# A Recommendation System as a Digital Marketing tool for Online Communities

**Federico Fiorini**

**School of Science**

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 15.9.2017

**Thesis supervisor:**

Prof. Petri Vuorimaa

**Thesis advisor:**

M.Sc. Xavi Magrinyà

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Federico Fiorini

Title: A Recommendation System as a Digital Marketing tool for Online Communities

Date: 15.9.2017 Language: English Number of pages: 6+61

Department of Computer Science

Professorship: T-111

Supervisor: Prof. Petri Vuorimaa

Advisor: M.Sc. Xavi Magrinyà

Recommender systems are able to predict users' preferences and items of interest, by analysing historical data on their behaviour and actions. Different techniques exist and are applicable in different scenarios. This thesis explores how to combine Content-Based and Collaborative-Filtering techniques in a hybrid system and how personalised recommendations and one-to-one marketing techniques can lead to an improvement in user engagement. Specifically, it is analysed the case of online platforms where there is no rating system in place. Results are empirically tested and evaluated with training/testing approach and recommendations seem to be quite accurate. However, further online evaluation is needed to measure any actual increase in user engagement.

Keywords: recommender systems, content-based, collaborative-filtering, digital marketing, one-to-one marketing, personalisation, user engagement

# Preface

I want to thank Professor Petri Vuorimaa for his good guidance and my instructor Xavi Magrinyà for his support.

I also want to thank my parents for being supportive in my choices regarding my studies abroad and my close friends for always being there when I needed them.

Otaniemi, 15.9.2017

Federico Fiorini

# Contents

# Symbols and abbreviations

## Abbreviations

| | |
|---|---|
| CB | Content-Based |
| CF | Collaborative-Filtering |
| IR | Information Retrieval |
| PCA | Principal Component Analysis |
| SVD | Singular Value Decomposition |
| BT | Behavioural Targeting |
| CPO | Chief Product Officer |
| NLP | Natural Language Processing |

# 1 Introduction

The concept of community has always been at the heart of the Internet since its first usages when scientists used to share data and collaborate in different projects and researches [1]. In fact, people with specific interests tend to join online groups and communities in order to interact extensively about the topic of interest, find relevant information, learn and discuss their hobbies with other people who share their same interests and goals. Those communities can be found on several online platforms, such as Facebook groups, Reddit, online forums and Slack, and they cover any possible interest, hobby or topic. Those topics vary from being really broad to being a niche interest. Communities are usually public, but can also be private and be accessible only by invitation. Usually, people find and join those communities when searching for information online about a specific hobby or interest. Many communities are also goal-oriented and have a common project or purpose that members are achieving together. Such communities usually exist already offline and members use online platforms to coordinate their projects and to discuss and communicate with the whole community.

Hubchat is a communication platform helping interest-based and purpose-driven communities accomplish together. Existing communities can create their own community space on Hubchat and users can also discover new communities based on their interests. Each community has a content feed with important posts shared by the users. Posts can contain text, links to external websites and pictures. Users can interact with other users' posts by *liking* them or leaving a *comment* to start a discussion. A deeper discussion between users is also possible thanks to the real-time chat channels and direct private messages.

Users who join several communities might not always be active in all of them and might not see important information or posts that they might find relevant and interesting. Also, posts and content shared in certain communities might be relevant for users who do not belong in that community. For these reasons, I decided to develop a *recommender system* that recommends content and posts to the users pooling from every public community but prioritising undiscovered content from the communities the users belong to. With such a system I firstly intend to make users come back to their communities and secondly discover new communities by recommending relevant content. In fact, such recommendation system is going to be used as a *digital marketing tool* in order to improve metrics on user retention and activation.

## 1.1 Problem statement and research questions

– The field of recommender systems is rich and several different approaches have been studied and implemented. Every method has different strengths and weaknesses and every use case needs to be assessed in order to decide

which approach to take. *What recommender system approach gives better recommendations in the case of online communities platforms where the main goal is to recommend relevant content to the users in order to make them come back to their communities and discover new ones?*

– When building a model to predict recommendations, it is necessary to know what items a user likes or dislikes. Usually, this is achieved through a rating system where users can rate items positively or negatively. However, when such rating system is not in place, it is not a trivial task to understand whether a user likes or not a specific item. *Considering the lack of a rating system, what features can be used in order to build a user profile that describes the user interests and the type of items that the user likes?*

– Recommendations and personalised content is surely an important feature to improve user experience by showing only relevant content and items of interests. This improves the general feeling of a user when using the platform. Nevertheless, user satisfaction is not the only benefit from implementing a recommender system. In fact, if a user is satisfied and finds the content relevant, he will come back to the platform to consume content and possibly generate revenue for the platform owner. Thus, *can this recommendation system be used as a tool for digital marketing in order to improve user retention and activation metrics?*

## 1.2 Research methods

In order to answer the research questions, different methodologies will be used. First of all, a literature review of different recommendation systems is needed in order to understand strengths, weaknesses and use cases of each method. Secondly, a qualitative research is necessary to understand better the current use case and to define what are the requirements and what exactly wants to be achieved. Data for this qualitative research will be gathered by interviews with the company representatives. After an analysis of such data, assumptions will be made on what recommendation system method to implement to get better results and on what features to use in order to build users and items profiles. A recommendation system model will be defined and an algorithm will be implemented. To put in practice the defined model, a build research methodology [2] will be used, first with the design of a software system and then with the implementation and deployment of part of it. Preliminary results will then be analysed in order to make conclusions.

## 1.3 Thesis structure

After this chapter of introduction, the thesis is structured as follows:

- Section 2: an extensive literature review on recommender systems is presented, starting with a definition of what a recommender system is, followed by an analysis of different algorithms and approaches. The section ends with a review of how recommender systems are used in digital marketing in order to implement one-to-one marketing strategies and personalization.
- Section 3: the Hubchat use case is deeply studied, qualitative data are analysed and the goals to be achieved with the implementation of a recommender system, are presented. After such analysis, it is proposed an approach to take.
- Section 4: the implementation of the proposed recommender system is explained in details.
- Section 5: the system is evaluated and preliminary results, problems and challenges are presented.
- Section 6: conclusions are taken.

# 2 Recommender systems

Recommender systems are software tools that predict the preferences of users in order to give them suggestions, such as what items to buy, what movies to watch or what books to read [3]. The main goal of such software tools is to create a list of recommended items for a specific user by aggregating and analysing user's previous actions, preferences and other users' suggestions. These personalised recommendations are usually expressed in the form of a ranked list of items [3]. User's preferences can be directly expressed, e.g., by a rating system, or inferred by indirect actions performed by the user, such as viewing a web page or watching a video for a time long enough to infer that the user likes it. Given the user's preferences, recommender systems compute the probability that she will be interested in specific products, in order to recommend the right items [4]. After the recommendations have been made, users can give implicit or explicit feedback on their accuracy and this feedback can be used to improve the next interactions of the user with the system [3].

The growth in the variety and amount of information and content accessible online and the rise of e-commerce websites with a multitude of different items, has led users to be overwhelmed by the amount of choice they had. Thus, having too many choices does not produce benefits, but rather decreases users' well-being [5]. In recent years, the usage of recommender systems has addressed this problem and decreased this information overload by making available to users only relevant content or items. Furthermore, recommender systems are very useful to the service providers, for many reasons. For example, in the case of e-commerce services, the main outcome of having a recommender system is the increasing of the number of items sold caused by relevant recommendations of items that the user likes and wants. Thus, this leads to an increase of the conversion rate: the number of users who consume an item over the number of simple views. Another main outcome of a recommender system is to provide to the user uncommon items that would be difficult to find without a precise recommendation [3].

When talking about recommender systems it is important to define what are the main objects to take in consideration. First of all, the *"items"* are the object of the recommendations and can be represented using different approaches depending on the type of item. *"Users"* are the people receiving the recommendations and information about them is usually stored in a user profile. Modelling a user profile is not a trivial task and the approach varies depending on the recommendation technique that has been chosen and from the specific use case. Lastly, the *"transactions"* are the interactions between the users and the system [3]. Transactions data are very important because the information retrieved is used by the recommender algorithm to generate recommendations.

Figure 1 shows an example of recommendations at Amazon.com. In this case, the recommendations are based on products previously purchased by the user. Furthermore, the *"user"* is the currently logged-in user, the *"items"* are the recommended

products and the *"transactions"* are the previously purchased products. Figure 2 shows an example of recommendations at YouTube. Also in this second example, the recommendations are based on videos previously watched by the user.

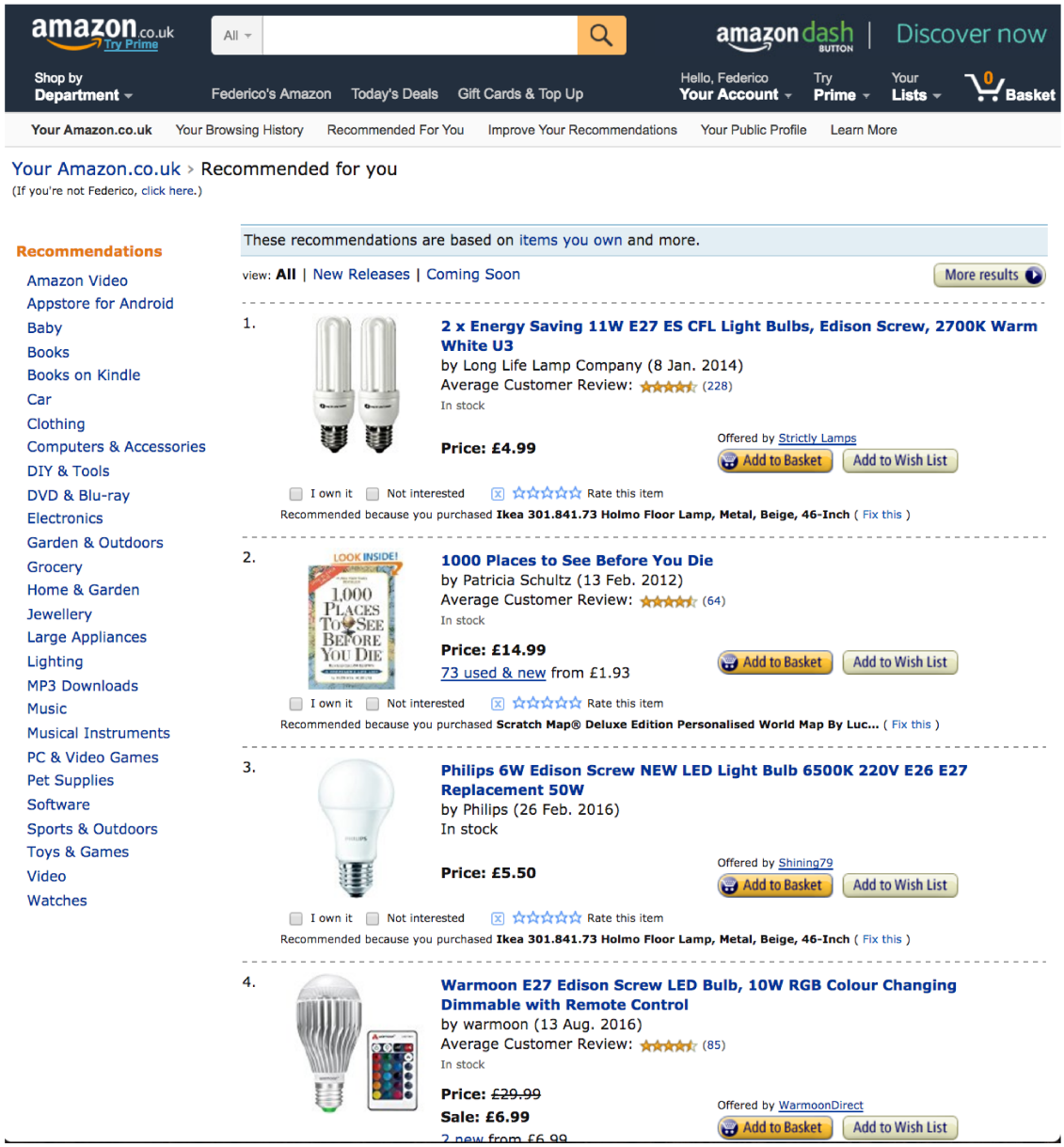


Figure 1: Amazon.com recommendations

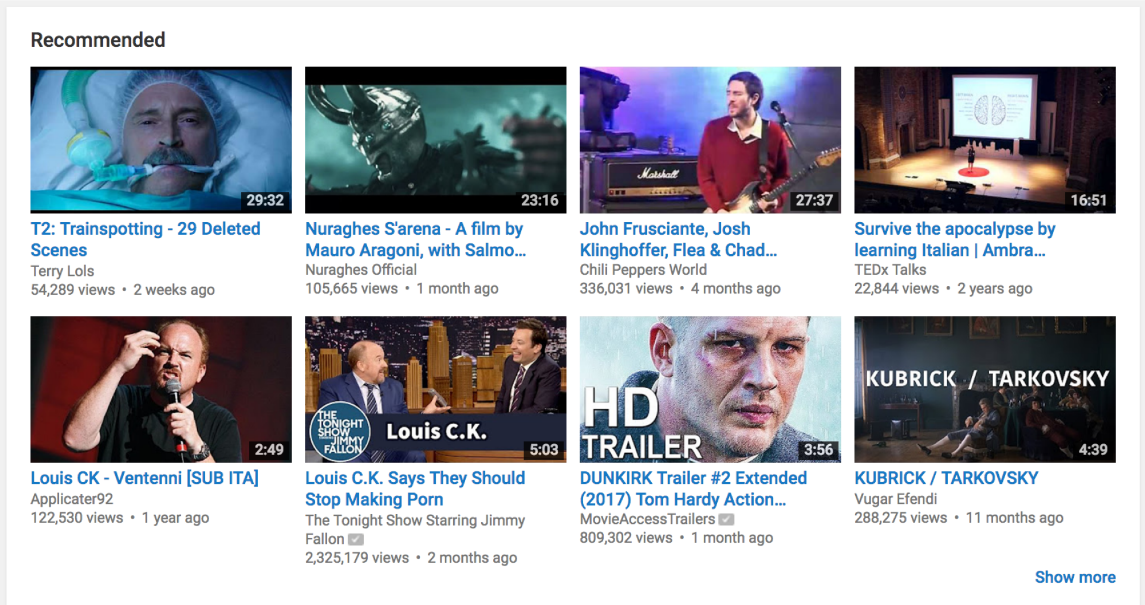


Figure 2: YouTube recommendations

## 2.1 Recommender algorithms

Recommender systems are a relatively new research field compared to classical systems and tools. In fact, they emerged as an independent research area only in the mid-1990s [6]. Nevertheless, in recent years it has become a very important research field. In fact, academic research has increased significantly in the last 15 years [4] with the outcome of several different solutions and methods being researched and implemented.

The idea behind recommendation systems is rather simple: people rely on recommendations provided by others in order to make decisions. To mimic this behaviour, the first recommendation systems recommended items to a user using recommendations provided by peers with similar interests [7]. This method is called Collaborative Filtering and it is one of the most common approaches to recommendations. Another well-known approach is the Content-Based method and the idea behind it is to analyse the content of an item in order to predict if the user will like it based on his preferences.

### 2.1.1 Content-Based

Content-Based (CB) recommendation systems recommend items to users based on items description and content that match user's profile and interests [8]. CB approach has its roots in the Information Retrieval (IR) field and uses many of the same techniques in order to extract features from the items. In fact, a user profile is

built by analysing the content of items that the user has rated or interacted with in the past. Thus, a pure CB recommender system bases its recommendations only on the match between the user profile and the new items' content [9].

Despite being used in several different domains, all Content-Based systems have some features in common. First of all, they all have a method to extract information from the items in order to describe them. Tools for text analysis are often used in case of documents. Items are usually represented by a feature vector or attribute profile. In order to replicate the human judgment of the importance of certain attributes, different weights are usually applied to different features [10]. Secondly, CB systems have a method to describe users and create a user profile based on previous interactions of the user with the items. Whether the user likes or not an item can be directly indicated by her actions, for example by rating it. Otherwise, this information can also be inferred by indirect actions, like having many interactions with the item, listening to the same song or watching a video many times. User profiles are also usually represented by a vector. Lastly, all CB recommender systems have a method to compare user profile and items content in order to produce a list of items to recommend. Different distance measures can be used to calculate the distance between the two vectors, such as Euclidean or cosine similarity. In the case of weighted features, the weight has to be considered when calculating the distance [10]. Figure 3 shows the cycle of a CB recommendation system: when a user likes or rates certain items, the user profile is updated, and new items that match the profile are recommended to the user.

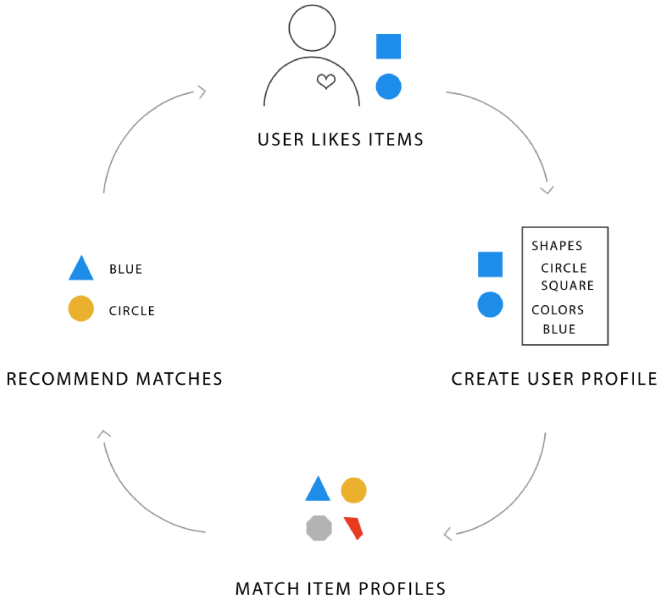


Figure 3: Content-Based recommendations cycle [11]

CB approach to recommendations has some main problems. First of all, it is not trivial to describe exactly what are the user's interests. In fact, since user profiles are built by analysing the previous interactions of a user with the items, if a user has not interacted with a certain type of items, it would be impossible to predict whether he likes or not an item of the same type. Furthermore, it takes a long time to build a rich user profile and in the case of new users, it is impossible to predict any recommendation [12]. Another problem with CB recommender systems is that it is impossible to produce recommendations if the information extracted from the content of the items is not enough to distinguish items that the user likes from items that she does not like [8].

### 2.1.2 Collaborative Filtering

Collaborative Filtering (CF) approach is known to be most successful recommendation technique [13] and it is widely used by several Internet companies such as Amazon, Netflix and Google News [14]. CF approach produces recommendations based on items ratings given by neighbour users who have similar profiles and interests. Unlike Content-Based approach, a pure CF system does not include any analysis of the items and recommendations are based only on similarities between users [9]. In fact, users are represented by an N-dimensional vector of items where N is the number of items available. The values of the vector are positive or negative depending on the rating given by the user to each item. The similarity between two users can be calculated with several methods, for example, measuring the cosine of the angle between the two vectors [16]. Recommendations are then selected from the items that similar users have purchased, liked or rated positively. To do so, different techniques can be used. A common method is to rank items based on how many similar users have liked them [15]. This approach can be classified as *user-based* CF recommendation system and it is the traditional method for collaborative filtering.

This traditional approach to CF comes with several challenges. First of all, it is computationally expensive since in the worst case scenario the time complexity is O(MN), where M is the number of users and N the number of items [15]. It is possible to optimise the algorithm to reduce complexity by sampling the users and by discarding very popular or unpopular items. Other dimensionality reduction techniques can be used, such as Principal Component Analysis (PCA) [17] or Singular Value Decomposition (SVD) [18]. However, all of these techniques have downsides and reduce the quality of recommendations in several ways [15]. Another challenge with traditional user-based CF systems is that, in order to build a rich user profile, it requires many interactions between the user and the items, such as purchases and ratings. In the case of a new user that has not interacted with many items yet, it is impossible to have a rich user profile, and thus to produce good recommendations. This is a well-known problem of collaborative filtering and it is usually referred as *cold start* problem [19].

A different approach to collaborative filtering is to take it as a classification problem using clustering algorithms to assign the user to the segment with the most similar users. Recommendations are then chosen from the pool of items that the users in the same segment have liked or purchased. In the classification phase, the user's vector is compared to the vectors that summarise the segments and the user is then assigned to the most similar segment. This approach is called *Cluster Model* [15] and it has better performance than traditional collaborative filtering because there is no need to measure similarities between each user but only between users and X number of vectors, where X is the number of segments. However, since every user in the same segment is considered similar in terms of recommendations, Cluster Model systems recommendations quality is low [20].

Amazon.com proposed a different approach to recommendations: *item-to-item* collaborative filtering [15]. In this approach, items-items similarities are identified and used to indirectly compute recommendations for the users [15, 21]. Instead of computing similarities between users and finding similar users, item-to-item approach matches each of the user's purchased item with similar items in order to find a list of neighbour items from where select the recommendations. Similarities between items are not calculated using their content, like in content-based approach, but rather considering items similar if they have been purchased together. A common method to measure this similarity is to represent each item with an M-vector, where M is the number of customers who have purchased or rated the item, and use the cosine of the angle between the two vectors as a similarity measure [15]. A similar-items table is created and then used when calculating recommendations. This approach to recommendations is very time consuming, in fact in the worst case scenario the time complexity is $O(N^2M)$. However, the complex and expensive part is to create the similar-items table and this computation is usually performed offline. Additionally, item-to-item CF approach addresses the cold start problem. In fact, this algorithm performs well also with limited user data and produces good quality recommendations even with users who have interacted with few items [15].

In general, collaborative filtering recommender systems solve many of the problems of using a pure content-based method. In fact, any type of items will be recommended regardless of whether the user has interacted or not with that type of items before. Furthermore, it is not necessary to analyse the item's content to extract information and build an item profile, nor to build a comprehensive user profile that describes the user's interests.

Figure 4 shows a comparison between user-based and item-to-item CF recommendations. In the user-based approach, one can see that users "green" and "blue" are similar because they liked common items. Thus, the items that user "green" has liked are recommended to user "blue". Instead, with the item-to-item approach, similarities are calculated between items based on which users liked them. One can see that "blue circle" and "blue square" are similar because they have both been liked by users "green" and "red". Thus, since user "blue" has liked "blue square", then "blue circle" will be recommended to him.

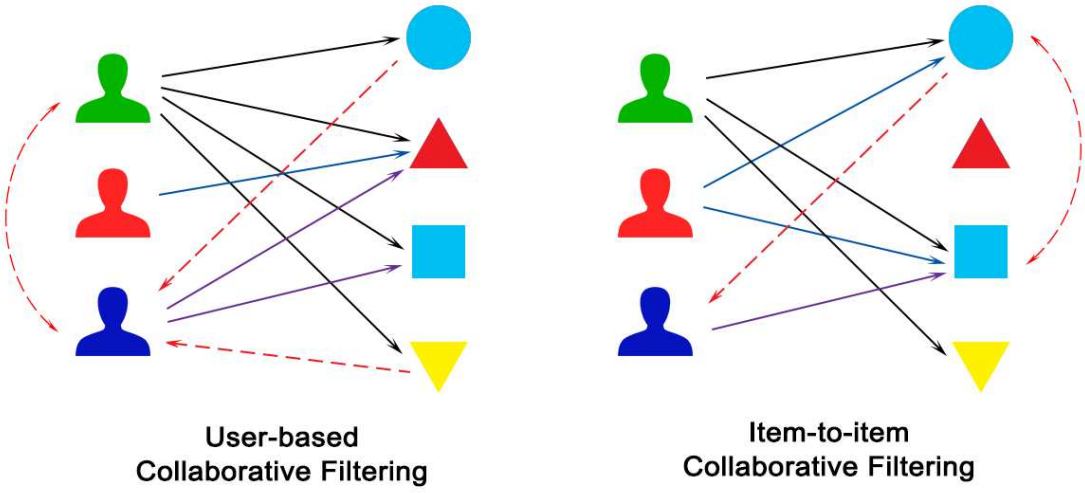


Figure 4: Collaborative-Filtering recommendations

### 2.1.3 Demographic

Demographic recommender systems base their recommendations on demographic information about the users, such as their language, age, gender or country [3], and aim to classify users in different categories and to produce recommendations based on demographic groups [22]. The idea behind demographic recommender systems is that different recommendations should be produced for different demographic classes [3]. This type of recommender systems have the benefit that do not require any history of user interactions with the items [22], so it is possible to produce recommendations for new users and there is no cold start problem.

However, demographic systems take a rather simplistic approach to recommendations and base their results on assumptions about demographic categorizations. Furthermore, these systems are not easily scalable and with an increase in the number of users there is a decrease in the coverage of the recommendations [23]. Thus, using this approach alone can only produce relatively accurate results. In fact, demographic systems are usually used in combination with other approaches, such as CF, to create a hybrid approach that takes into consideration demographic information about users.

### 2.1.4 Community-Based

The assumption behind community-based recommender systems is that people tend to rely more on recommendations made by their friends and by people they know. For this reason, community-based approach produces recommendations based on the preferences of the user's friends [3]. Communities are virtual user groups that

represent different user interests and are defined using clustering algorithms on the users [12]. The use of social network data makes it easier to acquire data about the user's social relationships, and thus to define the communities the user belongs to. Overall, recommendations based on social network data are not more accurate than traditional collaborative filtering recommendations [24]. However, they solve some of the problems of CF systems, such as the cold start problem. In fact, even if a user has not provided any ratings or purchased any items, recommendations can be made using his friends' ratings or purchases.

### 2.1.5 Context-Aware

Context-Aware recommender systems take into consideration contextual information when creating a user behaviour model and this improves predictions on his actions [25] and, thus, recommendations. In fact, users behaviour is influenced by the context in which it takes place [26]. Contextual information can be, for example, the period of the year, the time, location or the company of the other people the user is with.

There are mainly two approaches to include context in the recommendation process: *contextual pre-filtering* and *contextual post-filtering* [27]. In the pre-filtering approach, contextual information is used before performing the recommendations in order to filter out irrelevant transactions, i.e., ratings that are not relevant to the context. For example, purchases made in the summer might be filtered out if the recommendations are computed in the winter time. Thus, only the information that matches the current user context is used. With the post-filtering method, instead, contextual information is used after computing the recommendations to filter out irrelevant items from the output of the algorithm [28]. Different post-filtering approaches can be used for contextualising recommendations. One approach is to give a *weight* to the recommended items in order to reorder the recommendation list according to the relevance of the items with the current context. Another method is to *filter* out items that have low probability to be relevant with the context [28].

Studies [28] have proven that the pre-filtering method is a more reasonable solution because, in general, it provides better and quicker results. However, in certain cases, pre-filtering does not bring any improvements compared to an un-contextual recommender system. In such cases, post-filtering techniques are a better alternative.

### 2.1.6 Hybrid approach

Hybrid recommender systems are based on a combination of different approaches mentioned above. Using a combination of different methods, hybrid systems aim to take the advantages of each approach to fix their disadvantages. Many hybrid approaches have been studied and implemented [9, 12, 28] and most of the times

a hybrid approach is chosen over a pure recommender method. For example, [9] describes a hybrid content-based and collaborative filtering system that analyses items content in order to use this information to build user profiles, which are then directly compared to each other, in order to find similar neighbour users. Items are recommended both when they match with the user profile, and when they are purchased or highly rated by a neighbour user. Such approach reduces the weaknesses and limitations of pure collaborative filtering or content-based methods, as well as gaining their strength points. Furthermore, demographic and context information can also be added to the system in order to improve recommendations even further.

## 2.2 Recommendations in digital marketing

Recommendation systems are able to predict what items the users will find relevant, and this very valuable information can be applied to many sides of a business. In fact, besides improving the product or service that a company is providing, recommendations and personalised content can be used for marketing purposes in order to improve the effectiveness of advertisement or metrics such as conversion rate and user retention. Marketing is one of the key components of any successful business and it consists in "the activity of identifying, anticipating and satisfying customer requirements" [29]. Thus, the role of marketing is to contribute to increasing shareholder value "by developing relationships with valued customers and creating a competitive advantage" [30]. Over the years, marketing has evolved following the development of new technologies. In fact, when a new technology becomes popular and overpasses the early adopter phase, innovative marketers explore ways they can leverage the emerging technology to reach their target audience. After the technology becomes mainstream, it is integrated into standard marketing practices [31]. This is how marketing has evolved and passed from advertisement on paper to television to online and digital. In fact, with the rise of social media and social networks, marketers have found in the online marketing the most effective way to reach consumers. However, digital marketing has not only become a new channel for advertising, but rather a completely new marketing strategy which requires a different approach and understanding of customer behaviour and it includes many marketing techniques such as Search Engine Optimization (SEO), Search Engine Marketing (SEM), content marketing, social media marketing and data-driven marketing.

The evolving of technology did not provide to marketers only a new channel to reach consumers, but also many instruments to understand who their customers are. This led to the development of new marketing techniques such as Market Segmentation [32], i.e., dividing a market into several user groups, or segments, based on different characteristics such as age, gender, interests and location. By segmenting the consumers into groups that will respond similarly to marketing strategies, it is easier to personalise the marketing campaigns and make them more effective. Furthermore, by collecting behavioural information about users' online activity, it is possible to deliver specific advertising relevant to their interests. This technique is

named Behavioural Targeting (BT) [33] and it is used to improve the effectiveness of marketing campaigns.

### 2.2.1 Personalization and one-to-one marketing

As seen above, segmenting a market into several target groups allows marketers to apply different marketing strategies according to the group's characteristics, and thus to create more effective campaigns and have better results. However, segmentation might not be enough to have good results. In fact, even if done in a granular way using many features (e.g., gender, age and location) this technique implies that all people in the same group have similar interests, and thus treats them all in the same way. For these reasons, segmentation has been brought to the most granular level by personalizing marketing content at the level of the individual person. This strategy is called *one-to-one marketing* and it consists of increasing the precision of personalization to the level that every customer receives content that is the most relevant for her, and these predictions are based on the analysis of user data. Figure 5 shows that the difference between segmentation and one-to-one marketing is in the level of personalization. In fact, with no personalization at all, the same marketing strategy is applied to all the consumers and one can classify this as one-to-all marketing. With segmentation, we have a one-to-n marketing, where n is the number of segments, or groups, that the consumers have been divided in. Finally, with extreme personalization we can consider every segment to consist of one person only and in this case, one can talk about one-to-one marketing [34]. Thus, extremely personalised marketing delivers to an audience of one. This makes it a very effective way to perform marketing because the right content is delivered to the right consumer and, by consequence, users are more satisfied. However, this strategy relies on having accurate predictions of consumers interests. Thus, not accurate predictions can lead to misclassification of users and, by consequence, to poor marketing results.

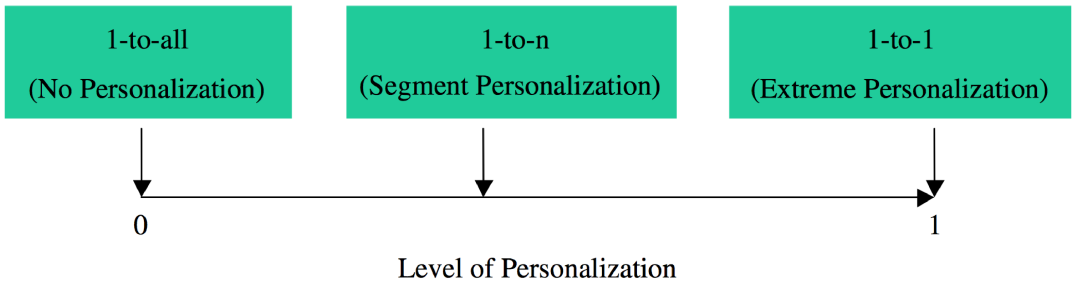


Figure 5: From 1-to-all to 1-to-1 marketing [34]

Four steps are necessary to implement one-to-one marketing: first of all, customers need to be identified in order to understand who they are; second, customers need to be differentiated; third, it is necessary to interact with them in order to start a relationship and acquire more user data; fourth, products and services have to be

personalised to fit users' needs [35]. By analysing users' data, recommender systems are able to understand their interests and predict what content they will find relevant. Thus, implementing a good recommender system is a key component to achieving effective personalization and good results in one-to-one marketing.

### 2.2.2 User activation, engagement and retention

Several measures are available to indicate the effectiveness of marketing campaigns. *User activation rate* measures the number of people who were activated by a campaign. For example, in the case of an email campaign, people who clicked on the provided URL are considered activated. Another important factor is *user engagement*, which can be measured by tracking *engagement actions* performed by the users. Such actions are specific to each product/business and have to be decided in each use case. Thus, engagement actions define whether the users are engaged or not. Several metrics can be used to measure the overall user engagement: DAU (Daily Active Users) and MAU (Monthly Active Users) measure the number of users who are active (i.e., users who performed at least one engagement action) within a specific amount of time. Furthermore, a high user engagement leads to *user retention* [36], which measures the percentage of users who come back to the product and perform an engagement action within a certain amount of time.

User activation, engagement and retention are very important marketing metrics and their improvement is crucial for any product. By using the results of a recommender system, it is possible for marketers to create campaigns targeted to each individual user, and thus increase the probability to engage the users with relevant content and make them perform specific engagement actions. Thus, recommender systems are a powerful tool for marketers who want to increase user engagement metrics by applying one-to-one marketing techniques.

# 3 Recommendations in a community-based platform

Hubchat is an online platform where users can join several online communities and connect with other people sharing their same interests and purposes, in order to accomplish something together. Users can create and manage their own communities and also discover new communities based on their interests. The content of each community consists of posts shared by the users, which can contain text, links to external websites and pictures. Users can interact with the content by liking the posts or leaving comments to start a discussion. Furthermore, users can belong to several communities, being able to see and interact with the content of all of them. Communities can be public or private. In the first case, they are visible through the search feature and anyone can decide to join those communities. In the latter case, they are hidden in the search results and discoverable only by a shareable URL. Thus, users need to ask a join request and the administrators of the communities have to approve the requests. A discovery feature is also present in the platform and it allows users to discover the most popular public communities.

Considering the details of the platform presented above, a recommendation system seemed to be needed in order to improve the user experience and the relevance of the content. However, before defining what approach to recommendations to take, it was important to understand the requirements of such system and what goals and outcomes we wanted to achieve with it. Thus, an interview with Hubchat CPO clarified the goals and requirements. The interview questions have been formulated in order to understand user activity on the platform. In particular, if users are usually active on the platform by posting new content or commenting other's posts, whether they join several communities and if they are usually active in all of them or not, how do they discover new communities and how do community owners promote their communities.

The main outcomes from the interview are:

- Users usually find Hubchat platform when searching online for information about a specific interest and find a community on Hubchat that is relevant to their search. Sometimes they find the community itself, other times they find a post that has been shared in the community.
- When the community or the post is relevant to people's interests, they usually register on the platform in order to join the community. Thus, the main source of organic user registration in the platform comes through online searches.
- Some people receive invitations by their friends to join a specific community, and thus to register on the platform. This is the second source of organic users.
- Once a user has joined one community, he/she usually joins a few more at the time of registration thanks to the discovery feature that suggests a few popular

communities.

- After registration, users usually do not come back to the platform unless they receive a notification by someone who mentioned them in a comment or in a post.
- Active users, who come back to Hubchat platform quite frequently, are usually active only in the first community they joined, which is the reason they register on the platform in the first place.
- Users who belong to several communities are not usually active in many of them.
- After joining a few communities during the registration phase thanks to the discovery feature, users usually do not search and join new communities.
- Users are usually not aware of communities with similar content to the ones they already belong. Those communities might be of interest to the users but it is difficult to find them unless they actively search for them.
- The discovery feature is the only sort of recommendation system already implemented. However, it is based only on finding the most popular communities among the ones similar to the communities that the user already belongs to. Furthermore, the similarity is purely based on community description and not on the actual content, i.e., posts and comments. This approach is currently not returning good results and this is probably because it is based on a wrong concept: the simple fact that a user belongs to a community does not always mean that he actually likes the content. In fact, in many cases users do not return to many of the communities they have initially joined.

Given the information retrieved from the interview, together with Hubchat CPO we decided that a better recommendation system needed to be implemented. The idea behind it is rather simple: recommending content (i.e., posts) to the users based on what other posts they have been engaging in the past. It can be assumed that if a user engaged with a post by liking or commenting it, then he would probably engage with similar posts that she has not seen yet. The level of engagement of a user with a post can be calculated by combining data about how many comments she has left or whether she liked it or not. Since there is no a rating system, the level of engagement can be used as a rate. Furthermore, posts to be recommended should be pooled from any public community and not only the ones that the user has already joined. In fact, when a user belongs to several communities, it is possible that he does not see important information or posts that he would, otherwise, find relevant and interesting. Also, posts and content shared in certain communities might be relevant for users who do not belong in that community. However, posts that the user has already seen should be removed from this pool of possible recommendations.

## 3.1 Goals

After reviewing the qualitative data and analysing Hubchat use case, it is clear that the goals to be achieved through the use of a recommender system are mainly two. Firstly, by recommending to the users relevant content that they have not seen yet, we aim to make them come back to the communities that they joined but where they are not frequently active. Secondly, by recommending content pooled from any public community, including those that the user does not belong to, we aim to improve the process of discovery of new communities that might be of interest to the user. In fact, if the user likes the suggested content, she will likely join the community where it is shared, in order to be able to comment the post and start a discussion. Furthermore, considering the Fear Of Missing Out theory [39], the user will join the community in order not to miss any relevant and interesting future content.

Generally, with the use of a recommender system we aim to improve two important metrics:

- User activation in the communities: a user is considered not-activated when he joined a community but has never performed any action in it, such as posting new content, liking and commenting existing content. By recommending relevant content, our goal is to activate the user.
- User retention: we aim to retain those users that stopped using the platform or stopped visiting certain communities, by suggesting them content that they will find interesting and relevant.

## 3.2 Proposed approach: a hybrid recommender system

After reviewing the literature about recommender systems, many approaches have been considered valid for Hubchat use case, especially Content-Based and Collaborative Filtering. Initially, Content-Based methodology seemed the most valid candidate. In fact, the idea of recommending content to the users matched perfectly with the idea behind Content-Based approach. With a method to extract information from the content of the posts, it is possible to create an item profile that describes the posts. Thus, by viewing at the interactions of the users with the posts, it is possible to then create a user profile that describes user's interests. Once the user profile is rich enough, it is possible to match it with new incoming posts as well as previously posted content. The level of matching between items and users would be used in order to rank items and build a recommendation list.

Although CB approach seems like a good candidate for Hubchat use case, it comes with some main problems that cannot be underestimated. First of all, it takes a long time to build a rich user profile that describes entirely the user's interests. With new registered users who do not have a history of activities on the platform, it would take a long time before the recommendation system would be able to recommend

them relevant content. Furthermore, the user profile would be biased by the fact that initially users join only a few communities and are active mostly in one. This means that the user profile will represent only a small part of the user interests. For this reasons, CB approach alone will not provide recommendations of type of items that the user has not seen yet and, thus, it will not allow us to achieve our goal of discoverability of new communities.

CF is the second approach that we considered to be appropriate for our use case. In fact, with CF we can overcome the fact that a user profile does not initially represent all the user's interests, and thus the recommender system would be able to recommend items from any community. In fact, recommendations would depend only on what content similar users have liked across all the communities. In order to define similar users, we could maintain a vector of items that a person has interacted with, and compare users' vectors to find the most similar ones. However, by using this approach we encounter another well-known problem: the cold start. In fact, users who recently joined and engaged only with few items, will have a very little item vector which will not give good results when calculating the distance with other vectors to look for similar users. To overcome this problem, we considered using an item-to-item collaborative filtering approach. Instead of maintaining item vectors for each user, we would maintain user vectors for each item and they would indicate which users engaged with the item. By comparing the distance between the vectors we would find similar items. Finally, when a user engages with an item, by liking or commenting the post, we would be able to create recommendations pooling from the set of similar posts. By using this approach we would be able to create recommendations even for a new user who engaged only with one post. However, we would still have the cold start problem for items. In fact, new items would not have a vector descriptive enough to find similarities, and thus they would not be recommended.

Every considered approach has some downsides that we wanted to avoid. For this reason, I designed a hybrid approach that includes both CB and CF methodologies. The designed system will have:

- A method to extract content from the posts in order to create an item-profile for each post. The profile will describe what topics and keywords are contained in the post.
- A method to create a user-profile based on what posts he interacted with. The user profile will contain information about what topics and keywords the user is interested in.
- Methods to calculate the similarity between an user-profile and an item-profile and, therefore, to create a list of content-based recommendations based on user-item similarities.
- A method to represent items by the users who engaged with them. This second item-profile will not consider the content of the items but will be entirely based

on what users have liked or commented the posts.

- A method to calculate similarities between the item-profiles in order to find items similar to each other.
- A method to create a list of collaborative filtering based recommendations pooling from the set of similar items.

Finally, by combining the outputs of CB and CF methods, the system will define the final list of posts to recommend to the user. This approach will mitigate the downsides of the two pure methods and will produce better results.

# 4 Implementation

The implementation of the designed recommender system is a rather complicated task that requires various steps.

- First of all, it is necessary to implement a rating system on which recommendations will be based. Such rating system is necessary to indicate which items a user likes or not. This very important information is at the base of our recommender system. Once the rating system will be defined, all the ratings will be stored in a database.

- Secondly, in order to implement the CB part of our hybrid system, we need to create for each post an item profile that describes what is its content. Since posts are composed of text and images, in order to extract content out of them it will be necessary to use Natural Language Processing (NLP) [37] techniques on text and Image Recognition [38] on images. Third party APIs will be used for the content-extraction task and a user profile will be stored in the database.

- By analysing the item profiles of the posts a user has rated, it will be possible to create a user profile that describes what topics, keywords and categories a user is interested in. The profile will also be stored in the database in order to be accessible for later use.

- The last part of the CB approach is to implement a method that compares users and items profiles and, according to the level of matching between the two, it predicts whether the user would like the post or not, and thus, the rate that she would give to it. By analysing the predicted ratings and picking the higher ones, it is possible to create CB recommendations.

- The second part of our hybrid approach consists in the item-to-item CF method. In order to implement it, it is necessary to create a second item profile that describes posts as a list of users who have interacted with them (i.e., users who have positive ratings) with their correspondent rate.

- It is necessary to implement a method to measure the similarity between item profiles according to how many users the profiles have in common and whether those users gave similar rates. Such method will be used to find similar items.

- Once the similarities between items are defined, it is possible to create CF recommendations by pooling from the list of posts that are similar to the ones that the user has high rated.

## 4.1 An indirect rating system

In the Hubchat platform, there is no direct rating system that users can use to give a rate to the posts that they have seen. However, ratings are a key component of the designed recommender system. Thus, an *indirect* rating system has to be implemented. The available features that indicate an interaction between a user and a post are:

- *Post seen*: it indicates that a user has seen a post by clicking on it.
- *Like*: a user can like a post.
- *Comments*: a user can leave none, one or more comments under a specific post.

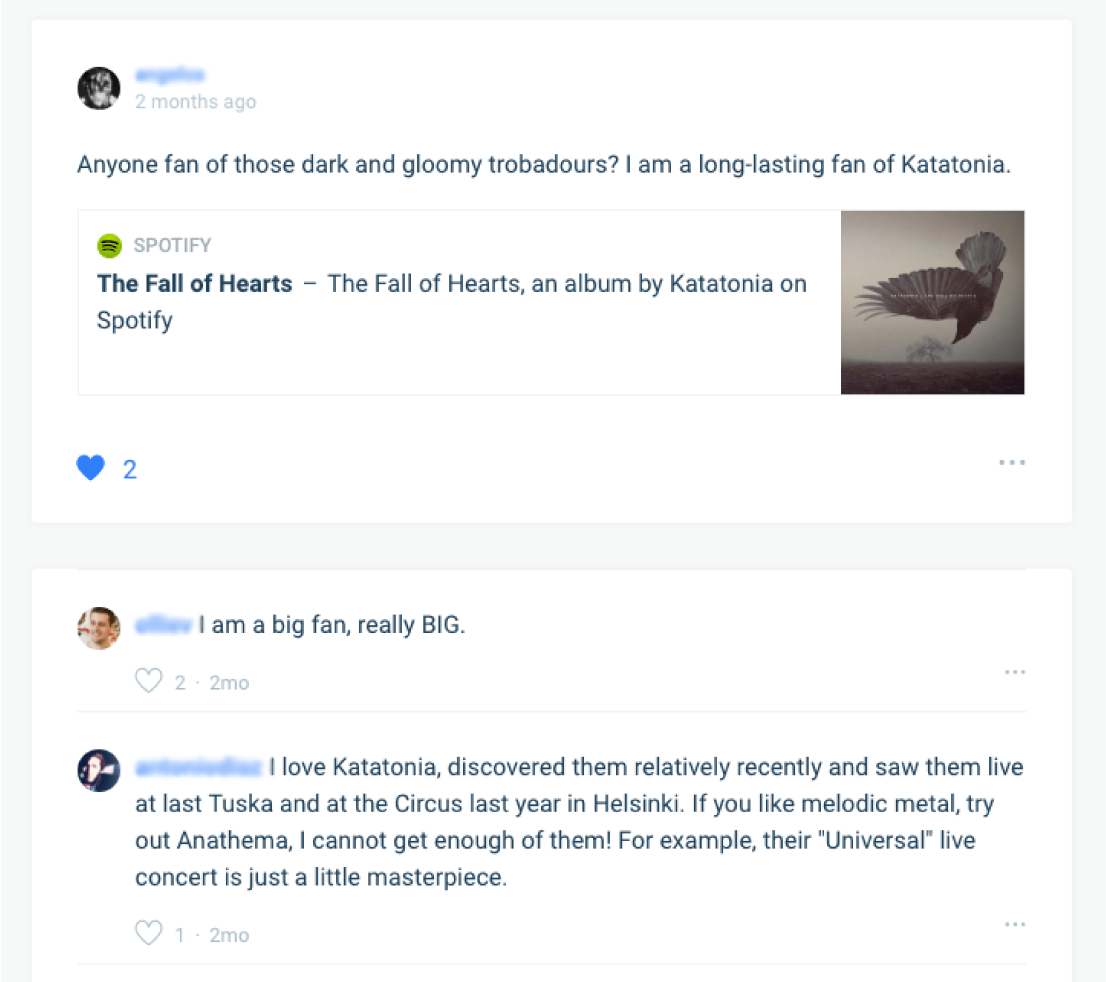


Figure 6: A post on Hubchat platform

Figure 6 shows how a post looks like. This particular post has two likes and two comments, which are the only features that users can use to interact with a post. Given the available features, only the *like* is a direct indicator that a user liked a

post. On the other hand, comments indicate that a user has *engaged* with a post and this is a very valuable information. In fact, with the recommender system, I do not necessarily intend to recommend posts that a user likes, but rather recommend posts that a user finds interesting, and thus interacts with. In fact, the *level of engagement* of a user with a post is the most valuable information that can be used to indicate whether she finds it relevant or not.

Given the current user interface and the available features, there is no direct indication of *dislike* of a post, in fact, there is no dislike button. To determine that a user finds a post irrelevant I use the information that she has seen the post but has not engaged with it. Thus, a low rate does not indicate that a user dislikes a post, but rather than she did not find it interesting and did not interact with it. If a user has not seen a post at all, then no rate is registered for it. Those posts missing a rate are going to create the pool of possible posts to be recommended by the recommender system.

### 4.1.1 The rating scale

In order to implement a rating system, a proper rating scale has to be chosen. The rating scale would indicate the *level of engagement* of a user with a post. At first, a classical 1-to-5 rating scale has been considered, where 1 means "unimportant" and 5 "very important". Thus, 1 and 2 are negative ratings, 4 and 5 are positive, and 3 is an average value, which means "neither important nor unimportant". However, given the fact that the only indication of a negative rate is when a user has seen a post but not engaged with it, it is difficult to estimate whether these cases should be rated as a 1 or a 2. In fact, a rating scale with two different negative rates is not easy to implement nor necessary. For those reasons, I opted for a 1-to-4 scale where 1 is the only negative rate and 2, 3 and 4 are increasingly positive rates. In Table 1, the rating scale has been represented in more details.

| *Rate* | *Meaning* |
|---|---|
| 1 | Unimportant |
| 2 | Slightly important |
| 3 | Somewhat important |
| 4 | Very important |

Table 1: 1-to-4 rating scale

### 4.1.2 Need for normalisation

At first, it seemed clear and simple that the way to determine the ratings would have been combining information about whether a user has liked or commented a post. In fact, it first appeared as a classification problem, where the combination of like/not like and the number of comments, would indicate what level of engagement the user had with the post and, thus, the indirect rate. Table 2 shows an example of classification.

| *Rate* | *Actions* | |
|---|---|---|
| 1 | No like & no comments | - |
| 2 | No like & 1-2 comments | Like |
| 3 | No like & 3-4 comments | Like & 1-2 comments |
| 4 | No like & 5+ comments | Like & 3+ comments |

Table 2: Example of classification to determine the indirect rate

With a classification method it would have been easy and immediate to determine the indirect rate of a post. However, results would have been wrong for various reasons. First of all, one can see in the given example in Table 2 that a like equates to two comments and it is translated to a rate of 2. Furthermore, four comments without like are translated into a 3, but five or more comments represent a 4. It is easy to see that there is a problem with this classification methodology: it is hard to determine how a combination of like/comments can be classified into the rating scale. Thus, this would require an extensive user study to understand what is the value of a *like* and of a *comment*. Furthermore, the values of likes and comments can be different for different users who have different behaviour. In fact, a user who tend to like posts very often might value a *like* differently than another user who likes only occasionally. The same can be applied to comments. For this reason, it is important to normalise the *likes* and *comments* according to each user's behaviour.

**Like rate** The *like rate* is the value that indicates what is the probability of a user to like a post that he has seen. It can be in the range between 0 and 1 and it can be calculated dividing the number of posts liked by a user over the total number of posts that she has seen.

$$Like\_rate = \frac{\#posts\_liked}{\#posts\_seen}$$

A high like rate indicates that a user tends to like most of the posts he sees. On the other hand, a user with a low like rate tends to like only a few of the posts that

he sees. Thus, the higher the like rate the lower is the value of a like for the user. Figure 7 shows the distribution of like rates across the users who have liked at least one post.

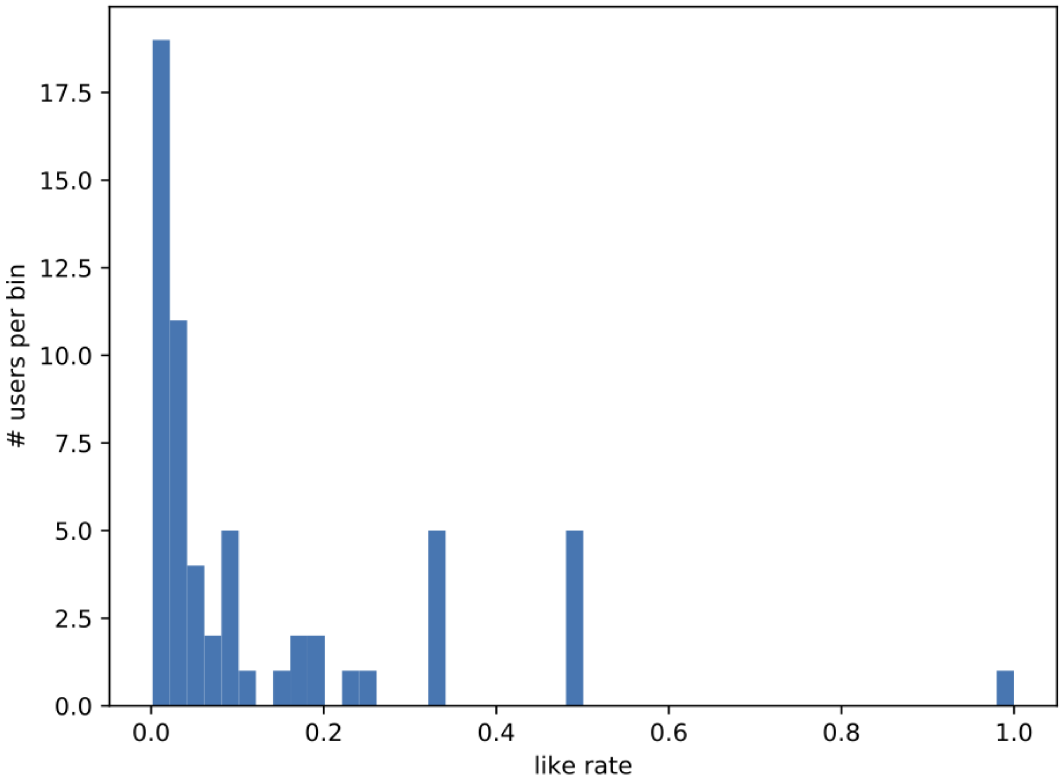


Figure 7: Like rate distribution

As showed by the histogram in Figure 7, the like rate of the users who have liked at least one post tends to be very low. In fact, most of the users have a like rate between 0 and 0.2. A minority of users have a rate between 0.2 and 0.5 and only a few users have a like rate of 1, meaning they liked every post they have seen.

Considering the fact that a *like* is, in any case, an indicator that the user found the post interesting, a like action should be translated into a positive rate, i.e., 2, 3 or 4. The like rate of the user can be used to determine which of the three possible positive rates is the most appropriate. Thus, a like action by a user with a high like rate will be translated into a 2 and the same like action by a user with a low like rate will be translated into a 4. Likes by users with an average like rate will be translated into a 3. Furthermore, it is important to determine what values are to be considered high, low or average rates. To do so, the area of the histogram has been divided into three segments of equal area and the values on the boundaries of the segments have been chosen as the limits between low, average and high rates. This approach assures a fair distribution of positive rates given the *like* actions. The results can be seen in Table 3.

| *Rate* | *Min range* | *Max range* |
|---|---|---|
| 2 | 0.0 | 0.02435 |
| 3 | 0.02435 | 0.09307 |
| 4 | 0.09307 | 1.0 |

Table 3: Like rate ranges

**Comment rate** The *comment rate* is the value that indicates what is the probability of a user to comment a post that he has seen. As the like rate, it can be in the range between 0 and 1 and it can be calculated dividing the number of posts commented by the user with at least one comment, over the number of posts that she has seen.

$$Comment\_rate = \frac{\#posts\_commented}{\#posts\_seen}$$

A high comment rate indicates that a user tends to comment most of the posts he sees, while a user with a low comment rate tends to comment only a few of the posts that he sees. Thus, the higher the comment rate the lower is the value of commenting a post for the user. Figure 8 shows the distribution of comment rates across the users who have commented at least one post. The histogram shows that the distribution of comment rates is denser between 0.0 and 0.5. There is also a relevant number of users with a comment rate of 1 that have commented every post they have seen.

In the same way as the *like* action, a *comment* indicates an engagement between the user and the post, and thus it is translated into a positive rate. Using the same approach used for the likes, the fact that a user has commented a post has been translated into rate 2, 3 or 4 depending on the user's comment rate: 2 for a high comment rate, 3 for an average rate and 4 for a low comment rate. Table 4 shows the ranges into which the rates fall. The ranges have been calculated with the same approach as for the like rate: dividing the histogram into three segments of equal area and using the values at the boundaries of the segments.

| *Rate* | *Min range* | *Max range* |
|---|---|---|
| 2 | 0.0 | 0.16666 |
| 3 | 0.16666 | 0.5 |
| 4 | 0.5 | 1.0 |

Table 4: Comment rate ranges

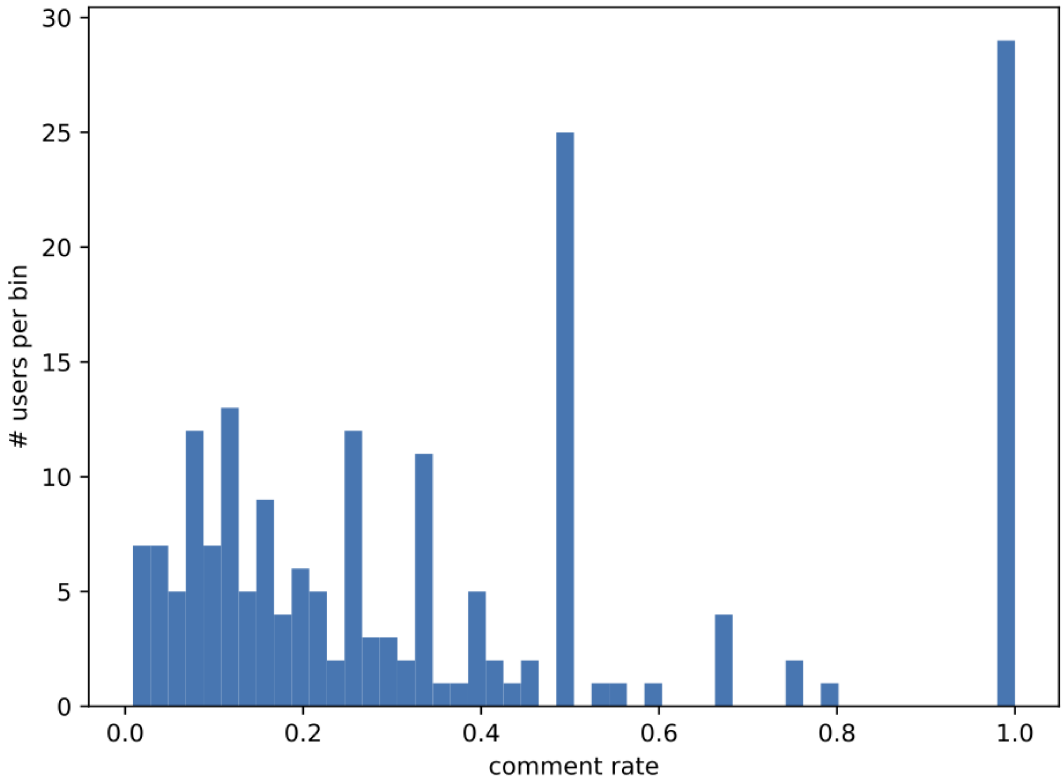


Figure 8: Comment rate distribution

**Comments per post average** Thanks to the like rate and comment rate, we have a rating system that translates user's actions into an indirect rate. Likes and comments have been interpreted as a binary feature: either a user liked or not a post and either she commented it or not. However, it is possible to extract more information by counting the number of comments left by a user to a post and comparing it to the average number of comments that she usually leaves when commenting a post. If the number of comments left on a post is significantly above the average, then its content is more relevant than others. Vice versa, if the number of comments is significantly below the average, then the content is less relevant and less engaging. This concept has been implemented into a modifier that corrects the base rate, given by the comment rate, by increasing it or decreasing it, depending on the comparison of the number of comments with the average value for that user. Table 5 shows the modifier values.

In order to implement the modifier, first of all, we need to calculate the average number of comments for each user. There are two options to calculate the average:

– Sum of comments *over* total number of posts seen

– Sum of comments *over* number of posts that have been commented

| *Modifier* | *# of comments* |
|---|---|
| -1 | Significantly below average |
| 0 | Average |
| +1 | Significantly above average |

Table 5: Rate modifier

The first option uses the total number of posts seen as the divider, while the second method uses only the number of posts that have been commented. Using the first option, the average is biased by the number of comments that have been seen but not commented, information already taken into consideration by the *comment rate*. Thus, in order not to take into consideration the comment rate twice, the average is calculated using the second method, i.e., over the number of posts that have been commented. In Figure 9, it is shown the histogram that represents the distribution of the average number of comments by user. It is clear that the majority of the users have an average of 1 to 3 comments per post.

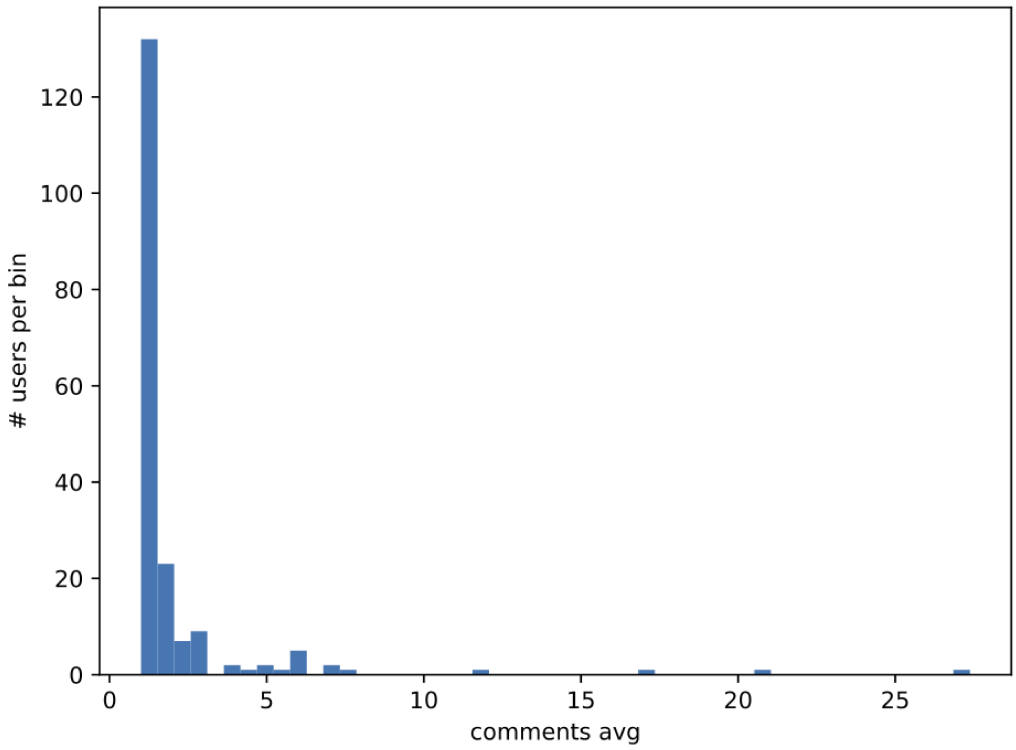


Figure 9: Number of comments average

In order to compare the current number of comments with the average value and calculate what modifier to apply, it is necessary to normalise the average with the scale of the modifier: -1 to +1. Figure 10 shows how this is done.


Figure 10: Modifier scale

Since the minimum number of comments is 0, the maximum value must be $avg*2$. In fact:

$$\frac{min + max}{2} = avg$$

$$\frac{0 + 2avg}{2} = avg$$

$$avg = avg$$

Given the function $f$, it is possible to normalize the number of comments into the modifier scale:

$$min\_scale = -1; max\_scale = +1$$

$$f(x) = \frac{x * (max\_scale - min\_scale)}{2avg} + min\_scale = \frac{2x}{2avg} - 1 = \frac{x}{avg} - 1$$

The result of the function is a real number between -1 and 1 that can be rounded to the closest integer value in order to obtain the final value of the modifier. Let us consider an example of a user that has an average number of comments of 3. If such user leaves three comments on a post, then the modifier will result as zero and will not change the base rate. However, if she leaves only one comment then the modifier will result in -1. In fact:

$$f(1) = \frac{1}{3} - 1 = -0.66$$

Rounding -0,66 to the closest integer we obtain the final value of -1.

In the case of a user leaving a number of comments greater than $avg * 2$, the final modifier will result in a number greater than +1. In these cases, the modifier to be applied will still be +1 because we consider it as the maximum value that the modifier can have.

**Final rate** Given the concepts described in the previous sections, the final rate of a post is calculated as the sum of the base rates and the modifier. The base rates are calculated using the like and comment rates, while the modifier using the number of comments. If the user has both commented and liked a post, the *like base rate* is considered as a modifier that corrects the *comment base rate* by +1, +2 or +3. The following algorithm can be applied:

**if** $comments > 0$ **then**
  $final\_rate \leftarrow comment\_base\_rate + comment\_avg\_modifier$
  **if** *liked* **then**
    $final\_rate \leftarrow final\_rate + like\_base\_rate - 1$
  **end if**
**else**
  $final\_rate \leftarrow like\_base\_rate$
**end if**

Since the maximum value in the rating scale is 4, a final rate greater than the maximum will anyway be recorded as a 4. This can happen in the cases of a user both liking and commenting a post. Figure 11 shows the distribution of final ratings. Most of the ratings are negative and indicate the posts that have been seen but the users have not interacted with. The positive rates are almost equally distributed between values 2, 3 and 4.

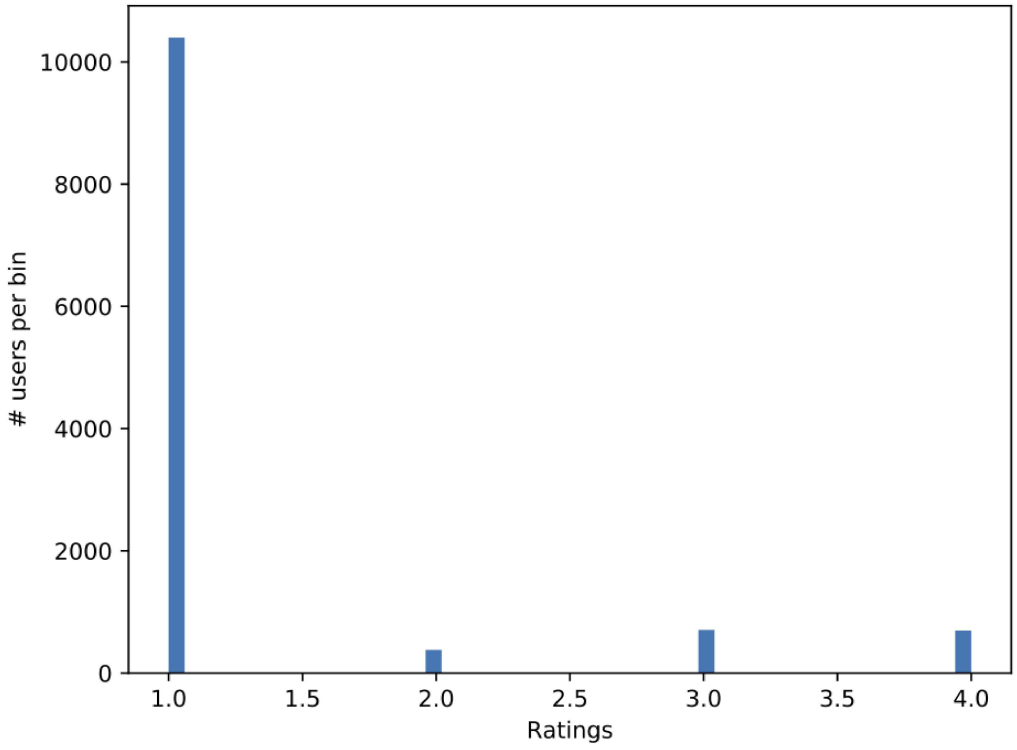


Figure 11: Final ratings distribution

## 4.2 Content-Based recommendations

The CB approach to recommendations requires multiple steps in order to be implemented:

- Building an item profile that describes what is the content of the post;
- Building a user profile that describes what are the topics that interest the user;
- Implementing a function that, given a user and an item profiles, predicts what is the rate that the user would give to such item;
- Recommend posts pooling from the items with a high predicted rate.

### 4.2.1 Item profile

An item profile has to describe, as detailed as possible, what is the content of a post. Since posts are composed of text, URLs and images, it is necessary to analyse them to extract meaningful information. The text contained in the web pages linked in the URLs is extracted in order to be analysed, as well as the text contained in the original post. NLP techniques are used to analyse such text. Instead, images are analysed with Image Recognition methods. The information extracted by such analysis is structured and it forms the item profile.

**Natural Language Processing** NLP is defined in [40] as a range of computational techniques for analysing and representing text for the purpose of achieving human-like language processing for a range of tasks or applications. Multiple methods can be used to accomplish certain type of language analysis. Nevertheless, the goal of NLP is to paraphrase an input text, describe its content, and thus produce a meaningful representation [40].

After considering various third party NLP APIs to be used to analyse the text contained in the posts, the choice has fallen upon IBM's commercial API: Watson Natural Language Understanding (NLU). In fact, Watson NLU is a comprehensive API that produces an advanced text analysis extracting semantic features such as categories, concepts, emotion, entities, keywords, metadata, relations, semantic roles, and sentiment [41]. In our use case, *keywords* and *categories* have been used to compose the item profile. In Figure 12 and 13, we can see the results of Watson API on the following text taken from a real post:

> *"History repeating. Some 20 years ago Bill Gates claimed that 640kB of memory is more than anyone will ever need. Now a Huawei exec claims that 4GB is more than enough for a smartphone. If it goes the same way, just as Moore predicted, I wonder what kind of wearable will use 25TB of memory. One thing I am sure, some people will think it is more than enough"*

| Hierarchy | Score |
|---|---|
| / science / social science / history | 0.58 |
| / technology and computing / consumer electronics / telephones / mobile phones / smart phones | 0.55 |
| / technology and computing | 0.25 |

Figure 12: Categories retrieved by Watson NLU API

| Text | Relevance |
|---|---|
| Huawei exec claims | 0.99 |
| History repeating | 0.78 |
| smartphone | 0.43 |
| memory | 0.43 |
| Gates | 0.41 |
| thing | 0.41 |
| kind | 0.40 |
| Moore | 0.39 |
| people | 0.36 |

Figure 13: Keywords retrieved by Watson NLU API

As shown in Figure 12, categories and subcategories are represented in hierarchical levels. To both categories and keywords, it is associated a score that represents the relevance of such category/keyword in the input text. The list of categories and keywords, with their relevance score, are stored in the database and make the post profile.

**Image Recognition** Image Recognition is the process of extracting meaningful information from an image by identifying and detecting objects and features in it. Thus, using Image Recognition techniques it is possible to classify images according

to their content. In order to have coherent results, I decided to analyse images using IBM's API: Watson Visual Recognition. In Figure 14, we can see an example of an image and the results of Watson API analysis on it: the list of *classes* that the image falls into and the corresponding score that represents how relevant that class is. Given the similar structure between classes extracted from the images and keywords extracted from the text, I decided to include classes under the keywords section of the post profile. Thus, the final item profile structure is composed of a *list of keywords and categories* with their *relevance score*.

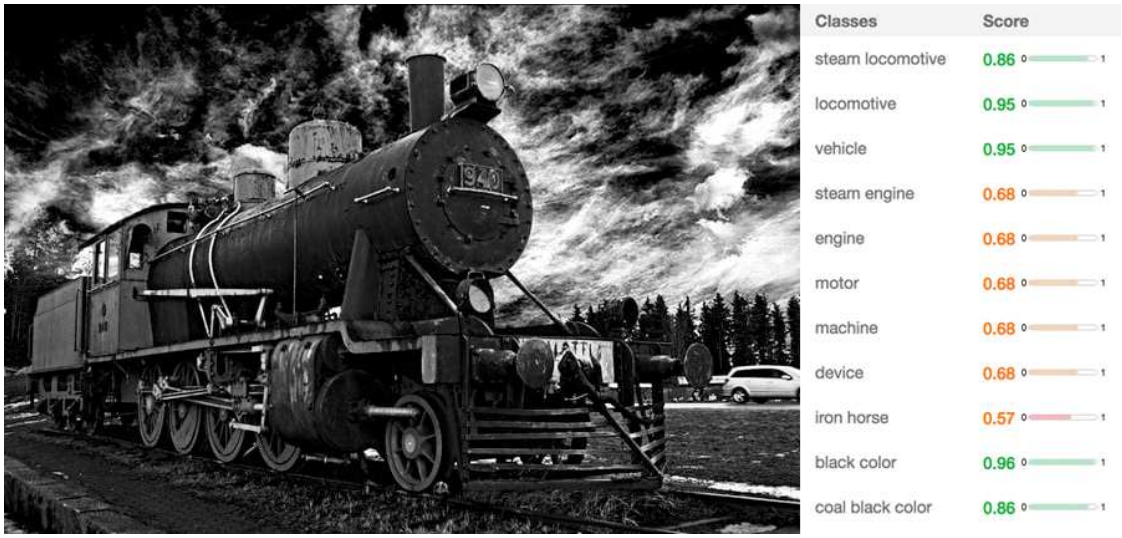


Figure 14: Classes retrieved by Watson Visual Recognition API

### 4.2.2 User profile

Building a user profile is an important step of the Content-Based approach. In fact, a rich profile can be used to predict what posts a user will like or not. Thus, a comprehensive user profile needs to contain information about what type of items the user likes or does not like. This information can be obtained by analysing the posts that the user has rated, i.e., post that he/she has seen and interacted with (positive rating) or seen and ignored (negative rating). For each of those posts, the post profile will be analysed. As seen in Section 4.2.1, the post profile (otherwise called item profile) is composed of a list of categories and a list of keywords and each category and keyword has a corresponding relevance score. For each category and keyword, it can be calculated the average rate weighted by the relevance. Follows a mathematical definition of the weighted average.

We define $C$ as the set of categories, $K$ the set of keywords and $P$ the set of posts rated by the user.

$$C = \{c_1, c_2, ..., c_n\}$$

$$K = \{k_1, k_2, ..., k_m\}$$

$$P = \{p_1, p_2, ..., p_i\}$$

We define $P_c$ as the set of posts containing the category $c$ and $P_k$ the posts containing the keyword $k$.

$$P_c = \{p | p \in P, p \ni c\}$$

$$P_k = \{p | p \in P, p \ni k\}$$

Keeping in mind that $rate_p$ is the rate given by the user to the post $p$ and $relevance_c$ and $relevance_k$ are numeric values between 0 and 1 that indicate how relevant a category/keyword is in the post, we define the weighted rate average of category $c$ and keyword $k$ as follows:

$$\bar{x}_c = \frac{\sum\limits_{p \in P_c} rate_p \times relevance_c}{\sum\limits_{p \in P_c} relevance_c}$$

$$\bar{x}_k = \frac{\sum\limits_{p \in P_k} rate_p \times relevance_k}{\sum\limits_{p \in P_k} relevance_k}$$

Finally, the user profile is composed of the list of all categories and keywords contained in the posts that he/she has rated, with their corresponding weighted rate average. Two versions of the user profile have been tried and tested:

- *Version 1*: the full profile containing all the categories and keywords found in the posts that the user has rated.
- *Version 2*: a profile built using only positive rates, i.e., greater than or equal to 2, and ignoring posts rated with a 1.

In version 1, there is no loss of information, thus the profile is richer and more accurate. However, given the high number of negative rates presents in the dataset, with version 2 I aimed to reduce the noise produced by the negative rates and keep a smaller profile that describes only what type of items the user likes. In Section 5, can be found a deeper discussion about the two versions and a comparison between the different results.

At first, the user profile is built using historical data of all the ratings given by a user and then stored in the database as a document containing a list of keywords and a list of categories, with their relative averaged rate. When new ratings are registered for the user, the user profile is updated.

### 4.2.3 Ratings predictions

In order to produce CB recommendations, it is necessary to predict what posts a user will interact with, i.e., predict the indirect rate given by the user to the post. Building item and user profiles was the foundation of the CB approach. In fact, by comparing item and user profiles it is possible to predict the rate given to the post, and thus what will be the level of engagement.

As described in Sections 4.2.1 and 4.2.2, the item profile is composed of a list of categories and keywords with their relevance score, i.e., a value between 0 and 1 that indicates how relevant that category/keyword is in the post. Also, the user profile is composed of a list of categories and keywords, each of them associated with a rate score: a value between 1 and 4 (2 to 4 in case of the user profile version 2, which includes only positive rates) that indicates the average rate given by the user to that category/keyword. By comparing keywords and categories present in both the user and item profiles, we are able to predict the final rate given to the post. However, keywords and categories are different by nature and format. In fact, in average the list of keywords is larger than the list of categories by one order of magnitude. For example, many posts belong in average to 3-4 categories but contain 30 to 40 keywords. Thus, this is reflected to the user profile where the order of magnitude of difference remains the same. Another difference is in the nature of categories and keywords: the first ones represent a class under which the posts fall into, e.g., the category "sport/football" indicates that the post is related to sport, in particular football. Instead, the latter ones indicate that a specific word is present in the text of the post with a certain relevance, e.g., the keyword "football" indicates that the word "football" is present in the text, but this does not necessarily mean that the post is related to football. For example, the post might be related to basketball and use the word "football" only to compare it to a different sport. A third difference is that keywords are specific words (or set of words) while categories contain many levels and are represented as a hierarchy of classes. In Table 6, we can see examples of keywords and categories taken from different posts. It is easy to see that keywords refer to specific concepts, objects or people, while categories can be either broad with only one root category, or more specific with many subcategories. Because of these differences, keywords and categories are treated differently when comparing item and user profiles.

| *Keywords* | *Categories* |
|---|---|
| "Samantha Carter" | "/sports" |
| "credit card" | "/sports/running" |
| "vehicle" | "/travel/tourist destinations/france" |

Table 6: Examples of keywords and categories

**Comparing keywords** When comparing keywords between item and user profiles, I extract the set of keywords present in both the profiles, i.e., the keywords that they have in common. Each keyword will have the rate taken by the user profile and the relevance score taken by the item profile. To predict the post rate, I calculate the average of all the keywords rates weighted by their relevance. Follows a mathematical definition.

We define $K_u$ and $K_i$ as the set of keywords contained in the profile of the user $u$ and of the item $i$ respectively.

$$K_u = \{k_1, k_2, ..., k_n\}$$

$$K_i = \{k_1, k_2, ..., k_m\}$$

We define $X_{ui}$ as the predicted rate given from the user $u$ to the item $i$ and it is calculated as follows:

$$\bar{x}_{ui} = \frac{\sum_{k \in (K_u \cap K_i)} rate_k \times relevance_k}{\sum_{k \in (K_u \cap K_i)} relevance_k}$$

The resulting rate is a value between 1 and 4 that will be used, together with the rate given by the categories, to predict the final post rate.

**Comparing categories** In order to compare categories, it is important to understand how they are structured. As seen in Table 6, categories are structured as a hierarchy composed of a main category and possible subcategories. Since there are overlaps between categories, they cannot be treated in the same way as keywords, that are, instead, unique. In fact, if the user profile contains the category "sports" and the item belongs to the category "sports/tennis", by using the same method used for keywords, they would be considered as completely different categories, and thus they would not be included in the calculation of the final rate. However, it is clear that they are similar categories and should be considered in the calculation. In fact, we can think of categories as trees where the main category is at the root and the subcategories are the children nodes. Figure 15 shows some example categories represented as trees.

When comparing categories present in the user and item profiles, it is necessary to handle the cases where there is overlapping between categories, i.e., categories who belong to the same tree, and thus have the same root. Two different methods have been implemented and tested.

– *Method 1*: the distance between two categories is used in the calculation. The distance is calculated by counting how many "edges" are between the two categories in the tree. For example, the distance between "travel" and

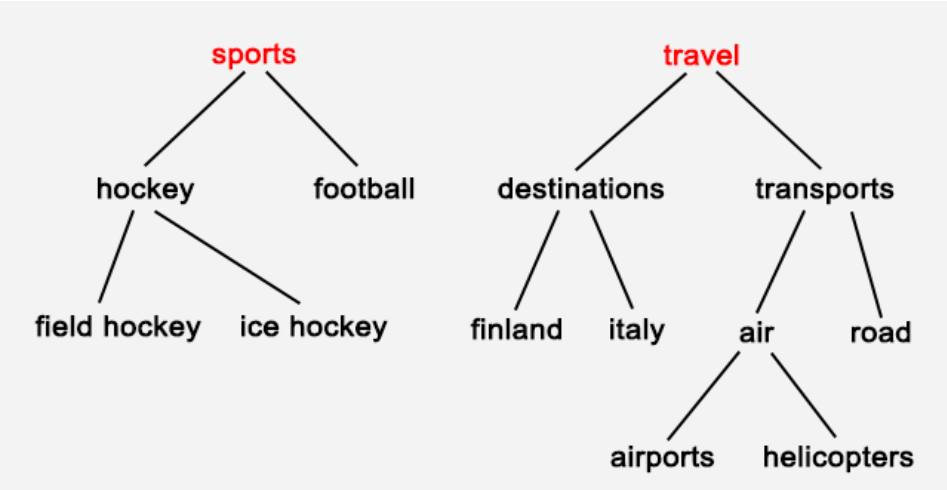


Figure 15: Categories represented as trees.

"travel/destinations" is 1 because one edge separates the two categories. By the same logic, the distance between "travel/destinations" and "travel/transports" is 2 and the distance between two equal categories is 0. When calculating the predicted rate, the distance between two categories is used to reduce the relevance of a category in the calculation. The following formula is used:

$$d = distance(cat_a, cat_b)$$

$$score\_cat_a = rate\_cat_a \times (relevance\_cat_b \times \tfrac{1}{2^d})$$

The distance is used as an exponential *multiplier* to reduce the relevance of the category. In fact, the higher is the distance the lower is the relevance: when the distance increases linearly, the relevance decreases exponentially. Thus, categories with higher distance will contribute less to the final rate. In fact, an exact match should be more relevant than a match between two categories that have the same root but are not equal.

– *Method 2*: the distance is not taken in consideration, instead the overlap between categories is used in the calculation. The overlap is calculated by counting how many subcategories two categories have in common. For example, the overlap between "travel" and "travel/destinations" is 1 because they have only the root category in common. By the same logic, the overlap between "travel/destinations" and "travel/destinations/italy" is 2, and the overlap between "travel/transports/air" and itself is 3. When calculating the predicted rate, the overlap is used to increase the relevance of the category in the calculation. The following formula is used:

$$o = overlap(cat_a, cat_b)$$

$$score\_cat_a = rate\_cat_a \times (relevance\_cat_b \times o)$$

The overlap is used as a *linear* multiplier to increase the category relevance. In fact, the higher is the overlap the higher is the relevance. Thus, categories with a higher overlap will contribute more to the final rate. In fact, categories with only one or two levels are more generic than categories with more levels, that are more specific. Furthermore, specific categories should have a higher weight because they express a more niche user interest. By including the overlap in the calculation, the level of specificity is taken into consideration.

To calculate the final post rate, each category in the user profile is compared to each category in the item profile that shares the same root. From each comparison, a score is calculated using one of the two methods seen above. By summing all the scores and dividing by the total relevance, we obtain a weighted average rate.

**Final predicted rate** The steps presented above produce two rates, which are calculated respectively by comparing keywords and by comparing categories. To produce a final rate, it is necessary to average the two results. At first, using the arithmetic mean [42] between the two rates seemed the easy and correct solution. However, given the considerations (already mentioned in Section 4.2.3) that the number of keywords and of categories differ by one order of magnitude, and furthermore that they have a different nature, it is clear that they should have different weight in the final rate. Thus, I decided to use a *weighted* arithmetic mean where the category rate has a higher weight. Different combinations of weight have been tried: 50-50 (same as the arithmetic mean, only for comparison purposes), 60-40, 70-30, 80-20. In Section 5, we can see a comparison between different combinations of weight. The following formulas are used to calculate the final predicted post rate.

$$rate_c = categories\_rate$$

$$rate_k = keywords\_rate$$

$$weight_c = categories\_weight$$

$$weight_k = keywords\_weight$$

$$rate = \frac{(rate_c \times weight_c) + (rate_k \times weight_k)}{weight_c + weight_k}$$

Once predicted the rates that a user would give to the posts he/she has not seen yet, we are able to produce CB recommendations by picking the posts with a high predicted rate.

## 4.3 Collaborative-Filtering recommendations

We have seen in Section 4.2 that the CB approach requires multiple steps in order to be implemented: analysing the content of the posts, creating an item profile that describes their content, creating a user profile according to which posts the user has high rated, predicting a rate between users and items profiles. Instead, the CF method approaches the recommendations problem from a different perspective, and thus requires less steps. Furthermore, in order to produce CF recommendations, it has been decided to use the item-to-item methodology [15], which consists of:

- Describing the items (i.e., the posts) by the users who interacted with them;
- Finding items similar to each other;
- Recommending items similar to the ones that the user has high rated.

### 4.3.1 Item profile

In the CB approach, we have seen how to describe posts by their content utilising different techniques, such as NLP and Image Analysis. However, with the CF method I take a different approach and the content of the posts is not taken into consideration. Instead, posts are described by the list of users who interacted with them and, more specifically, by the ratings given by those users. In fact, the profile consists of a *key-value* list where the keys represent the user and the values consist of the ratings given by them. Figure 16 shows an example of such item profile. Only positive ratings (i.e., greater than 1) are taken into consideration because we want to describe the post with the users who interacted with it and not with the users who ignored it.

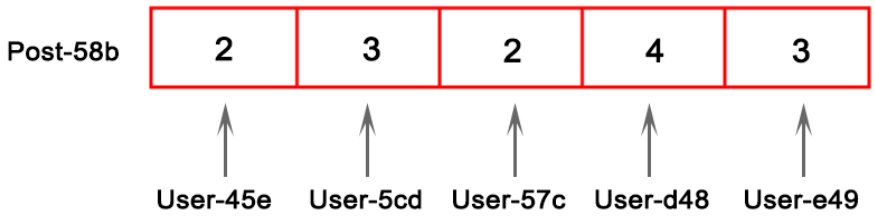


Figure 16: Post profile in Collaborative-Filtering.

### 4.3.2 Similarity

In order to find posts similar to each other, we need to compare item profiles and determine how similar they are. Since the item profile of a post consists of an N-dimensional vector where every user represents an independent dimension, we can represent it in a high dimensional vector space and use the angle between two

vectors as a measure of divergence between the vectors, and the cosine of the angle as the numeric similarity [43]. Thus, using the *cosine similarity* we can measure the similarity between two item profiles. Since all the ratings have positive values, the vectors are represented only in the positive quadrant of the vector space, and thus the angle between any pair of vectors is less than or equal to 90°. Thus, the cosine similarity is always a value between 0 and 1. The higher is the similarity, the more similar are the vectors, and thus the item profiles. The cosine similarity is calculated with the following formula:

Given two vectors A and B of dimension n

$$similarity(A, B) = \frac{A \cdot B}{||A||_2||B||_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The cosine similarity is calculated between every pair of posts that have received positive ratings. A similarity matrix is then built and stored in the database. Figure 17 shows an example of a similarity matrix. It is simple to notice that the cosine similarity between same posts is 1 because the vectors are exactly the same, and thus the angle is 0°.

| | **Post-58b** | **Post-8d3** | **Post-41h** | **Post-976** | **Post-4g1** |
|---|---|---|---|---|---|
| **Post-58b** | 1 | 0 | 0.8 | 0.2 | 0.5 |
| **Post-8d3** | 0 | 1 | 0.7 | 0.9 | 0.1 |
| **Post-41h** | 0.8 | 0.7 | 1 | 0 | 0.3 |
| **Post-976** | 0.2 | 0.9 | 0 | 1 | 0.9 |
| **Post-4g1** | 0.5 | 0.1 | 0.3 | 0.9 | 1 |

Figure 17: Similarity matrix.

### 4.3.3 Recommendation set

After analysing the item profiles and building a similarity matrix, it is possible to produce a set of CF recommendations for a specific user. In fact, by selecting the posts that the user has already high rated, and for each of them finding what are the most similar posts in the similarity matrix, we produce a set of posts similar to the high rated ones. By removing from this set the posts that the user has already seen and rated, it is possible to create a set of posts to be recommended. Figure 18 shows how the recommendation set is built.

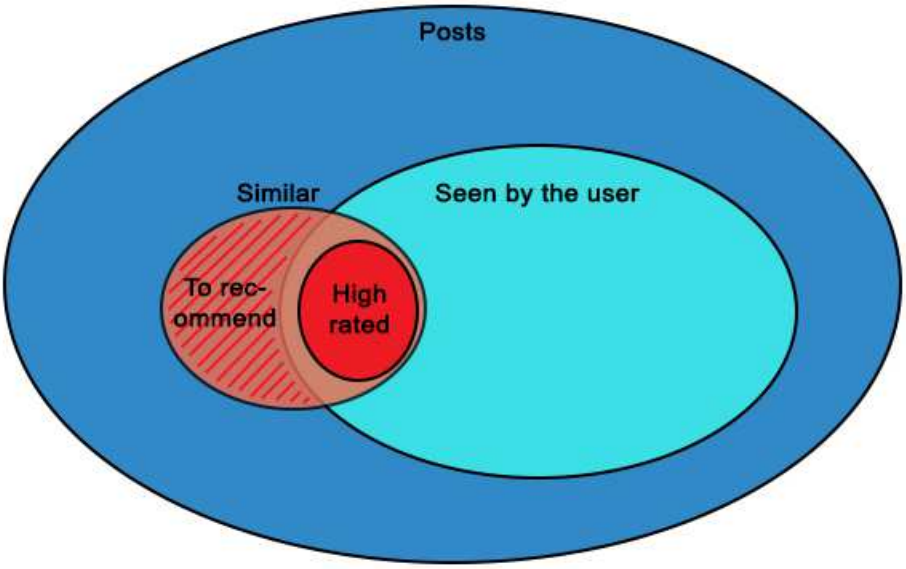


Figure 18: Recommendation set.

At first, this approach seems quite simple and straightforward. However, it is important to define the threshold of cosine similarity above which two posts are considered similar. To define the threshold I used the following formula:

$$threshold = avg(similarities) + \alpha \times \sigma(similarities)$$

Where $avg$ is the average value of the similarities, $\sigma$ is the standard deviation and $\alpha$ is a numeric value between 0 and 1. In Section 5, we can see how the results change with different values of $\alpha$, and thus different thresholds.

## 4.4 Hybrid recommendations

In Sections 4.2 and 4.3, we have seen the implementation of two separate recommender systems using two different approaches (CB and CF) and producing two separate recommendation sets. Given the different nature of the recommender systems, the two sets might differ in dimension and content. In order to combine the outputs of the two methodologies, and thus produce a final list of items to recommend, there are two possible options:

- *Union*: the two recommendation sets are merged into one unique set by combining all the elements and removing the duplicates.
- *Intersection*: only the items in common to both the recommendation sets are included in the final list of recommendations.

In the first case, the final recommendation list will be larger and include outputs from both methodologies. This ensures to take advantage of the strengths of both CB and CF approaches, and thus have a list of recommendation that covers most of the items that a user is likely to engage with. On the other hand, using the second approach will produce a shorter but more accurate recommendation list. In fact, the obtained results include only items recommended by both CB and CF approaches, and thus are more likely to be correct recommendations. However, this will exclude all those items that can be found only with one of the two approaches, and thus would not take full advantage of the strengths of both the methodologies. In Section 5, results from both *union* and *intersection* approaches are analysed and compared.

# 5 Evaluation

There are mainly two methods used to evaluate the results of a recommender system:

- *Online*: by exposing a group of users to the recommender system and asking direct feedback on the proposed recommendations.
- *Offline*: without any interactions with the users but only using historical data.

The online approach involves a direct feedback from the users, and thus it is more reliable in evaluating the performances of the recommender system and the user satisfaction. However, setting up such a system is hard, has a high cost and requires a large number of users willing to give feedback. Instead, the offline method is simpler to set up and faster to execute. In fact, it consists of using a part of the historical data to test the results of the system and analyse the quality of the predictions. Furthermore, it is easy to reproduce the same tests and compare different versions of a recommender system. Thus, I opted for an offline evaluation.

The dataset that I used is a subset of Hubchat database that covers the period between May 2016 and March 2017. The dataset contains 5,662 posts of which 3,311 have been rated. The users are 3,014 and in total, they produced 12,184 indirect ratings. However, only 1,784 of the given rates are positive.

Since the implementation of our recommender system involved using several parameters and different variants, it is possible to encounter the problem of overfitting [44]. Thus, in order to overcome this problem I decided to split the dataset into three different sets: *training*, *validation* and *testing* sets. The training and validation sets are used to estimate the values of the parameters: the training set is used as input to produce recommendations using different parameters and the results are compared to the validation set. Thus, the parameters that fit better the validation set (i.e., that produce better results) are chosen to be used in another round of training, which is done by using the training and validation sets as input and the testing set to measure the final performance. Using this approach, the parameters are chosen in order to fit the validation set as well as possible and the final measurement is performed on a set of data that has never been seen before from the system. Since the dataset is not very large (especially positive ratings are not many) I decided to include the majority of the data into the training set in order to produce a richer and more reliable model. Thus, I used the 10% of the data as the testing set and another 10% of the remaining data as the validation test, leaving the 81% of the full dataset for training. In Figure 19, one can see how the split has been done.

Another important consideration to make is whether to split the dataset *time-dependently* or not. In fact, a *time-independent* approach would split the dataset using a random order, and thus would produce a training set with ratings that might be more recent than some ratings in the validation and testing sets. This would produce results that are biased by the knowledge of user's future choices. Instead,

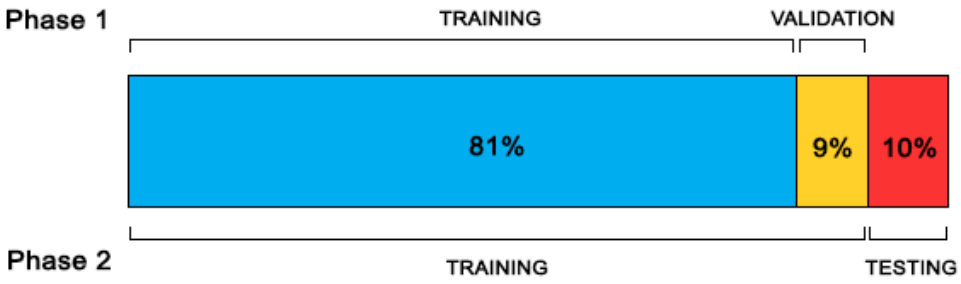


Figure 19: Dataset split in training, validation and testing sets.

using time as a splitting factor would make sure that all the ratings in the testing set are more recent than the ones in the training set. Thus, this approach replicates a real-world situation and gives more reliable results.

The last choice to be made is between splitting the dataset as a whole or using a *user-centered* split. In the first case, the entire dataset of all the users is separated into training, validation and testing sets. This split reflects perfectly a real-world situation. However, this can lead to have some users with ratings only in the training set and not in the validation and testing sets, or vice versa only in the testing set and not in the training. In such a scenario, it would not possible to properly evaluate the recommender system. In fact, the system cannot produce recommendations for users who are not present in the training set. Likewise, recommendations for users who are not present in the testing set cannot be evaluated because there are no ratings to be compared with. For this reason, I decided to use a user-centered split in which ratings are split for each user separately. This approach is not strictly time-dependent and does not perfectly reflect a real-world situation, but it enforces every user to have ratings in each of the sets, and thus it allows us to better evaluate the recommender system.

A different approach commonly used to evaluate performances is *k-fold cross validation*, in which the dataset is split into $k$ subsets (folders) of equal size. One folder is used as a testing set and the remaining $k$ - *1* folders are used as training set. The training/testing procedure is performed $k$ times, and each time a different folder is used as testing set and results are then averaged. This approach prevents the overfitting problem by testing results with many combinations of data. However, cross validation uses a random (and completely time-independent) split, and thus it is very distant from a real-world situation. For this reason I decided not to use this approach, but instead to split the dataset into training, validation and testing sets using a user-centered approach and time as a splitting factor.

## 5.1 Metrics

The most common method to evaluate the performance of a recommender system is to evaluate its accuracy and measure how close the recommendations are to the real preferences of the user [45]. However, in order to compare the accuracy of different systems, one or more metrics must be selected. This task is not trivial because there is a lack of standardisation and a large variety of different metrics. One set of common metrics are the *predictive accuracy metrics* that measure how distant the predicted ratings are from the real ratings [45]. Many error measure metrics fall into this category, such as Mean Absolute Error, Mean Squared Error and Root mean Squared Error. These metrics provide a good measurement in the cases where the recommender system predicts what would be the rates given by the users. However, in our case we do not have an exact prediction of a rate, but instead the algorithm classifies whether an item should be recommended or not, and finally it provides a list of items to be recommended. Thus, I decided to use a different category of metrics (*classification accuracy metrics*) that measure the accuracy with which the system makes correct decisions about whether to recommend an item or not [45].

### 5.1.1 Precision and Recall

As mentioned before, I decided to evaluate the recommender system with classification metrics. To evaluate performances in classification problems, it is necessary to first generate a confusion matrix. In Figure 20, one can see the confusion matrix used for a classification problem with two classes [46].

In our case, the two classes into which each item can be classified are *"to recommend"* and *"not to recommend"*. The confusion matrix can be filled by comparing the classes predicted by the recommender system with the actual true classes. More specifically:

- the *predicted positive* class is composed by the items that the system classified as "to recommend" (i.e., the final list of recommendations).
- the *predicted negative* class is composed by all the other items that have been classified as "not to recommend".
- the *true positive* class contains all the items that the user has high rated (with a rate higher or equal to 3).
- the *true negative* class contains all item that the user has low rated (with a rate lower or equal to 2).

From the confusion matrix we can read four important values:

- *True Positive (TP)*: the number of items that have been recommended by the system and that are also high rated for real by the user.

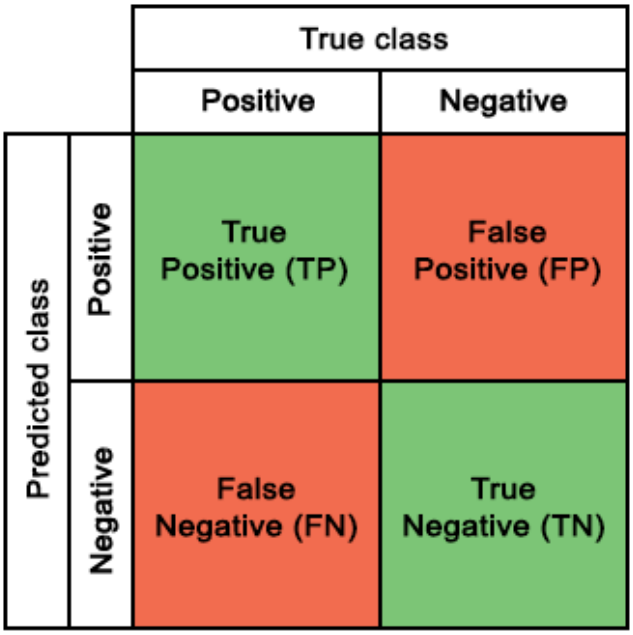


Figure 20: Confusion matrix.

- *False Positive (FP)*: the number of items that have been recommended by the system but do not have a high rate given by the user.
- *True Negative (TN)*: the number of items that have not been recommended by the system and, in fact, have a low grade given by the user.
- *False Negative (FN)*: the number of items that have not been recommended by the system but have a high grade given by the user.

With these values we can calculate three important metrics: *accuracy*, *precision* and *recall*.

- *Accuracy*: it measures the overall accuracy of the system calculated as a fraction of the correct classifications over the total number of items classified.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

- *Precision*: it indicates the fraction of recommended items that are indeed relevant to the user. It is calculated as a fraction of the correct recommendations over the total number of items recommended.

$$precision = \frac{TP}{TP + FP}$$

– *Recall*: it indicates the fraction of relevant items that are successfully recommended by the system. It is calculated as a fraction of the correct recommendations over the number of items that should have been recommended.

$$recall = \frac{TP}{TP + FN}$$

At first, accuracy might seem enough to evaluate the performance of the recommender system. However, measuring accuracy alone can lead to a bad evaluation of the system. In fact, accuracy can be easily increased by cheating the system. For example, given a dataset where TP are largely less than FP, by classifying every item as "negative" one can achieve a very high accuracy. On the other hand, with a dataset where TN are largely less than FN, one can achieve high accuracy by simply classifying every item as "positive". For these reasons, accuracy has not been chosen as a metric to measure the performance of our recommender system. Instead, *precision* and *recall* have been chosen as the measurement metrics because they complement each other and they give us more valuable information.

### 5.1.2 F-measure

As explained above, precision and recall give us different information about how well the algorithm has performed. However, in order to compare results from different recommender system variants, we needed to combine the two pieces of information into one final metric. For this purpose, I used the *F-measure* metric: a weighted harmonic mean between precision and recall. The F-measure can be calculated as follow:

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall}$$

$\beta$ is a positive real value that indicates the weights to give to precision and recall. For a value of $\beta$ equal to 1, the F-measure will be an harmonic mean between precision and recall. Other typical values of $\beta$ are 0.5, which weights precision higher than recall, and 2, which weights recall higher than precision. In our case, I choose a value of $\beta$ equal to 0.5 because I wanted to give more importance to precision rather than recall. In fact, the final list of items that I want to recommend is ideally short and precise because the goal is to periodically recommend few relevant posts to the user.

## 5.2 Parameters

In order to evaluate the performances of our recommender system, I decided to run multiple tests and compare results given with different combinations of parameters. Follows the full list of parameters that has been used.

CB parameters:

- *User profile version*: version 1 consists of the full profile containing all the categories and keywords found in the posts that the user has rated, while version 2 consists of a profile built using only positive rates and ignoring posts rated with a 1 (see Section 4.2.2)
- *Category comparison method*: method 1 uses the distance between two categories in the calculation, while method 2 uses the overlap (see Section 4.2.3 - Comparing categories)
- *Categories/Keywords weight*: the weight to give to categories and keywords when calculating the weighted average between the predicted rates (see Section 4.2.3 - Final predicted rate)

CF parameters:

- $\alpha$: the multiplier to apply to the function that defines the similarity threshold above which items are considered similar (see Section 4.3.3)

Hybrid approach parameters:

- *Merging method*: the methodology to use when merging recommendation sets produced by CB and CF algorithms.

Table 7 shows what values have been tested for each parameter.

| *Parameter* | *Values* |
|---|---|
| User profile version | 1 or 2 |
| Category comparison method | 1 or 2 |
| Categories/Keywords weight | 50-50 or 60-40 or 70-30 or 80-20 |
| $\alpha$ | 0 or 0.25 or 0.50 or 0.75 or 1 |
| Merging method | Union or Intersection |

Table 7: Parameters values

## 5.3 Training, validating and testing

In order to run tests accordingly to the dataset split, the following procedure has been followed:

- The dataset of ratings has been split into training, validation and testing sets;
- A similarity matrix (needed for the CF method) has been built using the

training set as input;

- For each user with ratings in the training set, two lists of CF and CB recommendations have been produced:
  - The CF recommendation list has been produced by selecting similar items from the previously created similarity matrix and removing those items that the user has already rated in the training set;
  - The CB recommendation list has been produced by firstly, creating a user profile using the ratings in the training set as input and secondly, comparing the user profile with the items in the validation set and for each of them predict a rate. If the predicted rate is greater or equal than 3, then the item was added to the CB recommendation list;
- The CF and CB sets have been merged together into a final set of recommendations;
- A confusion matrix has been built and F-score is calculated.

Firstly, some preliminary tests have been run by following this procedure and calculating F-score for each individual user and then averaging the results over the number of users. Since the list of parameters is relatively short, this procedure has been repeated for each combination of parameters to compare different results. Table 8 shows the results for each combination of parameters.

| $\alpha$ | *User profile version* | *Category comparison method* | *Categories/ Keywords weight* | *Merging method* | *F-score* |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 60-40 | union | ~0.146 |
| 0 | 1 | 1 | 70-30 | union | ~0.146 |
| 0 | 1 | 1 | 80-20 | union | ~0.146 |
| 0 | 1 | 2 | 60-40 | union | ~0.146 |
| **0** | **1** | **2** | **70-30** | **union** | **~0.147** |
| 0 | 1 | 2 | 80-20 | union | ~0.146 |
| 0 | 2 | 1 | 60-40 | union | ~0.104 |
| 0 | 2 | 1 | 70-30 | intersection | ~0.088 |
| 0 | 2 | 1 | 80-20 | intersection | ~0.098 |
| 0 | 2 | 2 | 60-40 | union | ~0.104 |
| 0 | 2 | 2 | 70-30 | intersection | ~0.089 |
| 0 | 2 | 2 | 80-20 | intersection | ~0.098 |

| | | | | | |
|---|---|---|---|---|---|
| 0.5 | 1 | 1 | 60-40 | union | ~0.126 |
| 0.5 | 1 | 1 | 70-30 | union | ~0.126 |
| 0.5 | 1 | 1 | 80-20 | union | ~0.126 |
| 0.5 | 1 | 2 | 60-40 | union | ~0.125 |
| 0.5 | 1 | 2 | 70-30 | union | ~0.126 |
| 0.5 | 1 | 2 | 80-20 | union | ~0.126 |
| 0.5 | 2 | 1 | 60-40 | union | ~0.081 |
| 0.5 | 2 | 1 | 70-30 | intersection | ~0.076 |
| 0.5 | 2 | 1 | 80-20 | intersection | ~0.084 |
| 0.5 | 2 | 2 | 60-40 | union | ~0.080 |
| 0.5 | 2 | 2 | 70-30 | intersection | ~0.076 |
| 0.5 | 2 | 2 | 80-20 | intersection | ~0.085 |
| 1 | 1 | 1 | 60-40 | union | ~0.099 |
| 1 | 1 | 1 | 70-30 | union | ~0.099 |
| 1 | 1 | 1 | 80-20 | union | ~0.099 |
| 1 | 1 | 2 | 60-40 | union | ~0.097 |
| 1 | 1 | 2 | 70-30 | union | ~0.099 |
| 1 | 1 | 2 | 80-20 | union | ~0.099 |
| 1 | 2 | 1 | 60-40 | union | ~0.064 |
| 1 | 2 | 1 | 70-30 | intersection | ~0.060 |
| 1 | 2 | 1 | 80-20 | intersection | ~0.064 |
| 1 | 2 | 2 | 60-40 | union | ~0.064 |
| 1 | 2 | 2 | 70-30 | intersection | ~0.060 |
| 1 | 2 | 2 | 80-20 | intersection | ~0.065 |

Table 8: Preliminary results.

Results show that the best combination of parameters that produces the highest averaged F-score is:

– User profile version = 1

- Category comparison method = 2
- Categories/Keywords weight = 70-30
- $\alpha = 0$
- Merging method = union

However, despite the overall F-score being the highest compared to the other results, the value was very low (~0.147). Thus, I decided that a further analysis of the results was necessary. In fact, I discovered that there was a high variance of F-score between different users. Thus, I tried to isolate the user variable that affected the F-score measure, and I assumed that such variable was the *count of positive ratings* given by the user. Therefore, I grouped users by this variable and averaged results. In Figure 21, one can see F-score as a function of positive ratings count. One can notice that there is a large difference between F-scores depending on the positive ratings count value. Especially, users with few positive ratings (less than 5) have a very low F-score. Given the current dataset where the majority of the users have few positive ratings, this affects largely the final averaged result.

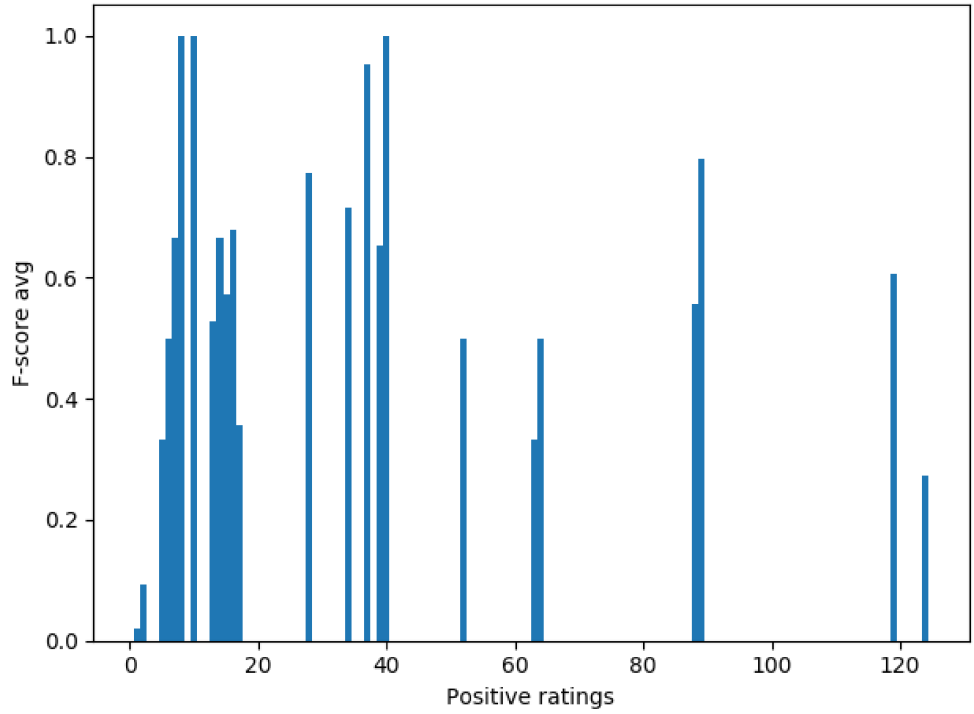


Figure 21: F-score values over positive ratings.

In order to overcome the problem of the large difference in results between users with few positive ratings (that we can consider as new users or not active users) and users with more positive ratings, the two following strategies have been applied:

- Group users in *bins* depending on their positive ratings, and find which com-

bination of parameters works better for each bin. In fact, this comes from the assumption that different combinations of parameters work differently depending on the count of positive ratings.

– Define a *threshold* of positive ratings count to consider a user eligible for recommendations, and thus do not consider users under that threshold when evaluating the results.

Given the preliminary results, I decided to group the users in bins with the ranges of positive ratings count shown in Table 9.

| *Bins ranges* | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 - 5 | 6 - 10 | 11 - 20 | 21 - 30 | 31 - 50 | 51 - 70 | 71 - 100 | 101 - 150 |

Table 9: Positive ratings ranges

The training procedure has been repeated with the binned version and results have been averaged for each bin instead of over the total number of users. The combination of parameters that produced the highest F-score has been chosen for each bin. Table 10 shows the results for each bin.

| *Bin* | *F-score* | *User profile version* | *Category comparison method* | *Categories/ Keywords weight* | $\alpha$ | *Merging method* |
|---|---|---|---|---|---|---|
| 1 - 5 | ~0.04 | 2 | indifferent | 60-40 | 0 | union |
| 6 - 10 | ~0.67 | 1 | indifferent | indifferent | 0 | union |
| 11 - 20 | ~0.44 | 2 | indifferent | 80-20 | 0 | intersection |
| 21 - 30 | ~0.52 | 1 | indifferent | indifferent | 0 | union |
| 31 - 50 | ~0.68 | 1 | indifferent | 70-30 | 0.25 | union |
| 51 - 70 | ~0.64 | 1 | 1 | 60-40 | 0.5 \| 0.75 | union |
| 71 - 100 | ~0.87 | 1 | indifferent | indifferent | 0.5 | union |
| 101 - 150 | ~0.54 | 1 | indifferent | indifferent | 0.5 \| 0.75 | union |

Table 10: Binned results.

Given the produced results, some considerations are necessary:

– The first bin (containing users who have 1 to 5 positive ratings) produces very low F-score results. This means that the chosen algorithm does not produce precise recommendations for users with few ratings (cold start problem [18]). Thus, this leads to the conclusion that the threshold for being eligible for recommendations is 6 positive ratings.

- The parameter alpha is directly correlated to the number of positive ratings. In fact, users with few positive ratings have better results with a low alpha value, while a higher value of alpha produces better results for users with more positive ratings. This can be explained by the fact that the value of alpha influences the cosine similarity threshold, above which two items are considered similar (in the CF recommendations). In fact, a high value of alpha produces a higher threshold, and thus less items are considered similar to each other. Users with few positive ratings need a lower threshold in order to include in the recommendation list most of the True Positive items. However, users with many positive ratings require to have a higher threshold in order to filter out from the recommendation list most of the False Positive items.

- Other parameters seem not to have any correlation to the number of positive ratings given by the user. More specifically:

  - "User profile version" and "merging method" have almost always the same value in every bin. Since I do not consider the first bin (1-5) as mentioned above, only one bin (11-20) has different results for the considered parameters. Thus, I assume that this an overfitting problem and I ignore the given results for the 11-20 bin.

  - For almost every bin, F-score results are equal regardless of the chosen "category comparison method". Only the 50-70 bin produces better results with the comparison method number 1. However, I consider this as an overfitting problem and ignore this result.

  - Different weight given to categories and keywords produces equal results in most of the bins, and thus the parameter value is indifferent. In some of the bins, a specific weight produces better results. However, I do not see any specific pattern nor correlation to the number of positive ratings and the resulted values might be again due to overfitting.

After tuning the parameters with the validation set and defining our model such that the best combination of parameters is chosen according to the number of positive ratings, a further round of training and testing was required. Thus, the training and validation sets have firstly been merged together into the new training set. Secondly, the procedure to produce recommendations has been repeated using the training set to produce the recommendation list and testing set to evaluate our results. Differently from the first round of training, the users with less than 6 positive ratings have not been considered eligible for recommendations, and thus have been ignored in the results calculation. Table 11 shows the resulting F-score measure calculated for each bin.

Results from the second round of training and testing (Table 11) are coherent with results from the first round (Table 10) with the only remarkable difference in the 6-10 bin, for which F-score calculated in the second round is considerably lower (~0.67 in the first round and 0.25 in the second). This difference can be attributed to

| *Bin* | *F-score* |
|---|---|
| 6 - 10 | ~0.25 |
| 11 - 20 | ~0.4341 |
| 21 - 30 | ~0.75 |
| 31 - 50 | ~0.8641 |
| 51 - 70 | ~0.6006 |
| 71 - 100 | ~0.6126 |
| 101 - 150 | ~0.6864 |

Table 11: F-score results after the second round of training and testing.

the fact that the 6-10 bin contains less users in the testing set than in the validation set. Problem and challenges related to the size of the database are discussed further in Section 5.5.

## 5.4 Analysis of the recommended items

The last necessary evaluation to be made was whether the designed recommendation system would accomplish the goal of bringing back users to their existing communities as well as discover new communities. In order to make this evaluation, recommended items have been analysed further to determine whether they belong to communities to which the user is subscribed already (*existing communities*), or to communities that the user has not discovered yet (*new communities*). Recommendation lists have been produced for all the users eligible for recommendations and recommended items have been analysed. In Table 12, one can see that, in average, half of the recommended items belongs to existing communities and the remaining half belongs to new communities.

| *Existing communities* | *New communities* |
|---|---|
| 49.8% | 50.2% |

Table 12: Distribution of recommended items among communities.

## 5.5 Problems and challenges

The evaluation of the recommendations has been made taking the data as they are and without taking into consideration certain biases. In fact, there are two major biases regarding users joining communities:

- At the moment of registration, users are suggested with a list of communities to join. Such list changes over time but is static and equal for every newly registered user. This means that users are biased when choosing communities to initially join and they do not join communities purely based on their interests.
- After the registration, it is difficult for the users to discover new communities based on their interests because of the lack of a discovery feature. In fact, communities are searchable only by name.

Given the poor search feature and the static list of suggested communities, it is clear that user's choice on what communities to join is biased and, furthermore, similar for many users. Thus, the list of posts that a user has interacted (or not) with is also biased. This could lead to misleading recommendations.

Another problem that I encountered is the size of the database. In fact, despite having a large enough list of users and posts, the number of actions (likes and comments) produced by the users is rather small. This affects the number of positive ratings, which is relatively small. Thus, one could produce and evaluate recommendations only for a small sample of users (the ones with enough positive ratings).

# 6 Conclusions

Recommender systems are very complex software systems that are composed of many components: a rating system that measures users preferences and that is based on direct or indirect user actions; items profiles that describe the items; users profiles that indicate users preferences; algorithms to compare profiles and find similarity measures; algorithms to predict user future preferences based on those similarities. Such a complex system can have many different variables and any component can be implemented with several different approaches. Thus, there is not a no-brainer solution that can be applied to any use case because every use case has different requirements and different desired outcomes. Some use cases are well-documented and several proposed approaches can be found in the literature. In fact, examples of e-commerce platforms and movie streaming platforms are common in the literature. The proposed approaches are usually based on a direct rating system in place on the platforms. Evaluations of such approaches are well documented and it is clear that the resulting recommendations improve sales performances. However, there is less research on different use cases where the requirements are different and where there is no direct rating system in place.

In this thesis, I described a different use case with a content-based community-based online platform where there are no items to sell to the users but only posts and communities, and where there is no rating system in place. Furthermore, I proved that a recommender system can be implemented also in such a use case. I designed and implemented an indirect rating system that measures the level of engagement of the users with the posts. I designed and implemented a hybrid recommender system that merges together results from two different approaches: CB and item-to-item CF. I evaluated results using Machine Learning techniques by splitting our dataset into training, validation and testing sets. I trained our model with training set, tuned several parameters with the validation set and evaluated the final results with the testing set. Finally, I listed evaluation methodologies that have been applied and I explained the metrics that have been used to measure the performance of our recommender system.

I can now answer the research questions that I asked at the beginning of this thesis:

*What recommender system approach gives better recommendations in the case of online communities platforms where the main goal is to recommend relevant content to the users in order to make them come back to their communities and discover new ones?*

I state that the approach that produces better recommendations is a hybrid recommender system that combines results from CB and item-to-item CF approaches. In fact, by merging results we are able to mitigate the downsides of the two pure approaches. More specifically, CB approach has a cold start problem for new users who do not have a large enough user profile to describe their interests, and thus

poor recommendations are produced for them. On the other hand, item-to-item CF approach has a cold start problem for new items that have not been seen and indirectly rated by many users, and thus do not get included in recommendations because of a low similarity measure. Thus, by merging the results of the two approaches, CB results mitigate the cold start problem of item-to-item CF and, vice versa, CF results mitigate the cold start problem of CB. Furthermore, such hybrid system has been implemented in a way that content to be recommended equally belongs to the communities that the user has already joined and to the ones that she has not joined yet. Thus, this approach meets the goals of making the users come back to their communities as well as discovering new ones.

*Considering the lack of a rating system, what features can be used in order to build a user profile that describes the user interests and the type of items that the user likes?*

Information about what items a user likes are necessary to develop any type of recommendation system. Thus, given the lack of a direct rating system in place, I chose to use an indirect rating system that measures the *level of engagement* of the users with the items and is based on user actions that indicate engagement. Such user actions consist of: post seen, liked and/or commented by the user. User actions are then translated into a rating scale by taking into consideration measures such as like rate and comment rate. The ratings are then used to build users and items profiles. Therefore, the features that have been used to indicate user interests and to build user profiles are user actions that indicate the level of engagement with the posts.

*Can this recommendation system be used as a tool for digital marketing in order to improve user retention and activation metrics?*

Marketing has always been evolving following developments and improvements in technology. Nowadays, digital marketing techniques, involving personalisation of content at the individual level, are possible thanks to the improvements in recommendation systems. Thus, recommender systems are definitely a tool that enables marketers to apply one-to-one marketing strategies. However, results in marketing metrics are correlated and directly proportional to results in recommendations. In fact, bad recommendations would lead to deteriorations of marketing metrics such as user retention and activation. Furthermore, marketing results will also depend on the ability of marketers to deliver recommendations with the correct timing and with effective channels.

Given the positive results achieved by the hybrid recommender system that has been implemented, I state that improvements in marketing metrics are possible if recommendations are delivered to the users in an effective way. However, I have no tangible results to support this assumption, which is only based on the evaluation of recommendation results and on knowledge of digital marketing.

## 6.1 Future work

Further work is needed in order to deploy the recommender system on the platform, deliver recommendations with several channels (such as emails and in-app notifications) and evaluate results. A/B testing is suggested in order to enable recommendations only to a small sample of users and compare user retention and activation metrics with the users who do not receive recommendations.

Improvements in the recommendation model are also possible with further research. In fact, as I detected that the count of positive ratings is a variable that influences the alpha parameter, other user variables can be identified and used to tune other parameters differently.

# References


[1] Armstrong, Arthur, and John Hagel. *The real value of online communities.* Knowledge and communities 74.3 (2000): 85-95.

[2] Amaral, J.N., 2011. *About computing science research methodology.*

[3] Francesco Ricci and Lior Rokach and Bracha Shapira, *Introduction to Recommender Systems Handbook*, Recommender Systems Handbook, Springer, 2011, pp. 1-35

[4] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, Jae Kyeong Kim, *A literature review and classification of recommender systems research*, Expert Systems with Applications, Volume 39, Issue 11, 1 September 2012, Pages 10059-10072

[5] Schwartz, B.: *The Paradox of Choice.* ECCO, New York (2004)

[6] Resnick, P., Iakovou, N., Sushak, M., Bergstrom, P., Riedl, J. (1994). *GroupLens: An open architecture for collaborative filtering of netnews.* Computer Supported Cooperative Work Conf.

[7] Mahmood, T., Ricci, F.: *Improving recommender systems with adaptive conversational strategies.* In: C. Cattuto, G. Ruffo, F. Menczer (eds.) Hypertext, pp. 73–82. ACM (2009)

[8] Pazzani M.J., Billsus D., 2007. *Content-Based Recommendation Systems.* In: Brusilovsky P., Kobsa A., Nejdl W. (eds) The Adaptive Web. Lecture Notes in Computer Science, vol 4321. Springer, Berlin, Heidelberg

[9] Marko Balabanović and Yoav Shoham. 1997. *Fab: content-based, collaborative recommendation.* Commun. ACM 40, 3 (March 1997), 66-72.

[10] Souvik Debnath, Niloy Ganguly, and Pabitra Mitra. 2008. *Feature weighting in content based recommendation system using social network analysis.* In Proceedings of the 17th international conference on World Wide Web (WWW '08). ACM, New York, NY, USA, 1041-1042.

[11] Steeg, F. van. Context-aware recommender systems. MS thesis. 2015.

[12] J. Kamahara, T. Asakawa, S. Shimojo and H. Miyahara, *A Community-Based Recommendation System to Reveal Unexpected Interests*, 11th International Multimedia Modelling Conference, 2005, pp. 433-438.

[13] Jae Kyeong Kim, Hyea Kyeong Kim, Hee Young Oh, Young U. Ryu, *A group recommendation system for online communities*, International Journal of Information Management, Volume 30, Issue 3, June 2010, Pages 212-219.

[14] Zhou Y., Wilkinson D., Schreiber R., Pan R. (2008) *Large-Scale Parallel*

*Collaborative Filtering for the Netflix Prize*. In: Fleischer R., Xu J. (eds) Algorithmic Aspects in Information and Management. AAIM 2008. Lecture Notes in Computer Science, vol 5034. Springer, Berlin, Heidelberg

[15] G. Linden, B. Smith and J. York, *Amazon.com recommendations: item-to-item collaborative filtering*, in IEEE Internet Computing, vol. 7, no. 1, pp. 76-80, Jan/Feb 2003.

[16] B.M. Sarwarm et al., *Analysis of Recommendation Algorithms for E-Commerce*, ACM Conf. Electronic Commerce, ACM Press, 2000, pp.158-167.

[17] Dohyun Kim, Bong-Jin Yum, *Collaborative filtering based on iterative principal component analysis*, Expert Systems with Applications, Volume 28, Issue 4, May 2005, Pages 823-830.

[18] Sheng Zhang, Weihong Wang, J. Ford, F. Makedon and J. Pearlman, *Using singular value decomposition approximation for collaborative filtering*, Seventh IEEE International Conference on E-Commerce Technology (CEC'05), 2005, pp. 257-264.

[19] Zhao, Xiaoxue. *Cold-Start Collaborative Filtering*. Diss. University College London, 2016.

[20] J.B. Schafer, J.A. Konstan, and J. Reidl, *E-Commerce Recommendation Applications*, Data Mining and Knowledge Discovery, Kluwer Academic, 2001, pp. 115-153.

[21] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. *Item-based collaborative filtering recommendation algorithms*. In Proceedings of the 10th international conference on World Wide Web (WWW '01). ACM, New York, NY, USA, 285-295.

[22] Burke, R. User Model User-Adap Inter (2002) 12: 331.

[23] Ghazanfar, Mustansar Ali, and Adam Prugel-Bennett. "A scalable, accurate hybrid recommender system." Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on. IEEE, 2010.

[24] Golbeck, J.: *Generating predictive movie recommendations from trust in social networks*. In: Trust Management, 4th International Conference, iTrust 2006, Pisa, Italy, May 16-19, 2006, Proceedings, pp. 93–104 (2006).

[25] Palmisano, C., Tuzhilin, A., and Gorgoglione, M. 2008. *Using Context to Improve Predictive Models of Customers in Personalization Applications*. IEEE TKDE, 20(11).

[26] Bettman, J.R., Luce, M.F, and Payne, J.W. 1998. *Constructive consumer choice processes*. JCR, 25(3).

[27] Adomavicius, G., and Tuzhilin, A. 2008. *Context-Aware Recommender Systems*. ACM RecSys Tutorial, 2008.

[28] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. 2009. *Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems*. In Proceedings of the third ACM conference on Recommender systems (RecSys '09). ACM, New York, NY, USA, 265-268.

[29] Drucker, Peter (1954). *The practice of management*. New York: Harper and Row Publishers.

[30] Paliwoda, Stanley J.; John K. Ryans (2008). *Back to first principles*. International Marketing: Modern and Classic Papers (1st ed.). p. 25. Retrieved 2009-10-15.

[31] Ryan, Damian. *Understanding digital marketing: marketing strategies for engaging the digital generation*. Kogan Page Publishers, 2016.

[32] Wedel, Michel, and Wagner A. Kamakura. *Market segmentation: Conceptual and methodological foundations*. Vol. 8. Springer Science & Business Media, 2012.

[33] Yan, Jun, et al. *How much can behavioral targeting help online advertising?*. Proceedings of the 18th international conference on World wide web. ACM, 2009.

[34] Arora, Neeraj, et al. *Putting one-to-one marketing to work: Personalization, customization, and choice*. Marketing Letters 19.3-4 (2008): 305.

[35] Weng, Sung-Shun, and Mei-Ju Liu. *Feature-based recommendations for one-to-one marketing*. Expert Systems with Applications 26.4 (2004): 493-508.

[36] Ahmad, Rizal, and Francis Buttle. "Customer retention: a potentially potent marketing management strategy." Journal of strategic marketing 9.1 (2001): 29-45.

[37] Chowdhury, Gobinda G. "Natural language processing." Annual review of information science and technology 37.1 (2003): 51-89.

[38] Javidi, Bahram. Image recognition and classification: algorithms, systems, and applications. CRC Press, 2002.

[39] Andrew K. Przybylski, Kou Murayama, Cody R. DeHaan, Valerie Gladwell, *Motivational, emotional, and behavioral correlates of fear of missing out*, Computers in Human Behavior, Volume 29, Issue 4, July 2013, Pages 1841-1848

[40] Liddy, Elizabeth D. "Natural language processing." (2001).

[41] IBM Watson Natural Language Understanding https://www.ibm.com/watson/developercloud/doc/natural-language-understanding

[42] Jacobs, Harold R. Mathematics: A human endeavor. Macmillan, 1994.

[43] Singhal, Amit. "Modern information retrieval: A brief overview." IEEE Data Eng. Bull. 24.4 (2001): 35-43.

[44] Hawkins, Douglas M. "The problem of overfitting." Journal of chemical information and computer sciences 44.1 (2004): 1-12.

[45] Herlocker, Jonathan L., et al. "Evaluating collaborative filtering recommender systems." ACM Transactions on Information Systems (TOIS) 22.1 (2004): 5-53.

[46] Delen, Dursun, and D. Olson. "Advanced data mining techniques." Springer. doi 10 (2008).